DOCTOR OF PHILOSOPHY

Semantic Segmentation of Terrain and Road Terrain for Advanced Driver Assistance Systems

Gheorghe, Ionut Valentin

Award date:
2015

Awarding institution:
Coventry University

Semantic Segmentation of Terrain and Road Terrain for Advanced Driver Assistance Systems

Gheorghe, I. V.

Semantic Segmentation of Terrain and Road Terrain for Advanced Driver Assistance Systems

By

Ionut Valentin Gheorghe

September 2015

Semantic Segmentation of Terrain and Road Terrain for Advanced Driver Assistance Systems

By

Ionut Valentin Gheorghe

September 2015

A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy

Acknowledgements

I would like to express sincere gratitude to my first academic supervisor Professor Weidong Li for his mentorship and patience. His advice and critique have helped me navigate well through the difficulties of this project. This gratitude extends to my second academic supervisor Professor Keith Burnham for his invaluable comments and to Olivier Haas for sparking my interest in image processing during one of his taught courses.

I would also like to take this opportunity to thank our industrial collaborators Thomas Popham and Anna Gaszczak for their support and guidance. Crucial to this thesis, they have provided a dataset and I have benefited a great deal from their industrial and past academic expertise. Many of the project milestones have been set as a result of our ongoing consultation.

Special thanks go to my friends and colleagues Walid Allafi and Zhonghua Shen for making the research office a very pleasant and familiar environment.

Last but not least, I would like to thank my family for their unconditional support and encouragements throughout this long journey.

Abstract

Modern automobiles and particularly those with off-road lineage possess subsystems that can be configured to better negotiate certain terrain types. Different terrain classes amount to different adherence (or surface grip) and compressibility properties that impact vehicle manoeuvrability and should therefore incur a tailored throttle response, suspension stiffness and so on. This thesis explores prospective terrain recognition for an anticipating terrain response driver assistance system. Recognition of terrain and road terrain is cast as a semantic segmentation task whereby forward driving images or point clouds are pre-segmented into atomic units and subsequently classified. Terrain classes are typically of amorphous spatial extent containing homogenous or granularly repetitive patterns. For this reason, colour and texture appearance is the saliency of choice for monocular vision. In this work, colour, texture and surface saliency of atomic units are obtained with a bag-of-features approach. Five terrain classes are considered, namely grass, dirt, gravel, shrubs and tarmac. Since colour can be ambiguous among terrain classes such as dirt and gravel, several texture flavours are explored with scalar and structured output learning in a bid to devise an appropriate visual terrain saliency and predictor combination. Texture variants are obtained using local binary patters (LBP), filter responses (or textons) and dense key-point descriptors with daisy. Learning algorithms tested include support vector machine (SVM), random forest (RF) and logistic regression (LR) as scalar predictors while a conditional random field (CRF) is used for structured output learning. The latter encourages smooth labelling by incorporating the prior knowledge that neighbouring segments with similar saliency are likely segments of the same class. Once a suitable texture

representation is devised the attention is shifted from monocular vision to stereo vision. Surface saliency from reconstructed point clouds can be used to enhance terrain recognition. Previous superpixels span corresponding supervoxels in real world coordinates and two surface saliency variants are proposed and tested with all predictors: one using the height coordinates of point clouds and the other using fast point feature histograms (FPFH). Upon realisation that road recognition and terrain recognition can be assumed as equivalent problems in urban environments, the top most accurate models consisting of CRFs are augmented with compositional high order pattern potentials (CHOPP). This leads to models that are able to strike a good balance between smooth local labelling and global road shape. For urban environments the label set is restricted to road and non-road (or equivalently tarmac and non-tarmac). Experiments are conducted using a proprietary terrain dataset and a public road evaluation dataset.

Keywords

Advanced driver assistance systems, terrain recognition, semantic segmentation, monocular vision, stereo vision, machine learning, superpixels, supervoxels.

Contents

List of Figures

List of Tables

List of Abbreviations

ADAS	Advanced driver assistance systems
AP	Average precision
CAN	Controller area network
CHOPP	Compositional high order pattern potentials
CNN	Convolutional neural network
CRF	Conditional random field
DARPA	Defence advanced research projects agency
FN	False negatives
FNR	False negative rate
FP	False positives
FPFH	Fast point feature histograms
FPGA	Field programmable gate array
FPR	False positive rate
GLOC	Global and local
GMM	Gaussian mixture model
GSP	Global shape prior
GT	Ground truth
IR	Infra-Red
JLR	Jaguar Land Rover
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
KL	Kullback-Leibler divergence
LAGR	Learning applied to ground robots
LBP	Local binary patterns
LIDAR	Light radar
LR	Logistic regression
MCMC	Markov chain Monte Carlo
MRF	Markov random field
pb	Probability of boundary
PFH	Point feature histograms
PRE	Precision
RBF	Radial basis function
RBM	Restricted Boltzmann machine
REC	Recall
RF	Random forest
RGB	Red, green and blue colour channels
RGBD	Red, green, blue and depth channels
ROI	Region of interest
SAD	Sum of absolute differences
SCRFFPFHGSP	Superpixels under a conditional random field with (saliency from) fast point feature histograms and global shape prior
SCRFHGSP	Superpixels under a conditional random field with (saliency from) height and global shape prior
SIFT	Scale invariant feature transform
SLIC	Simple linear iterative clustering
sp	Superpixels
SSVM	Structural support vector machine
SVM	Support vector machine
TN	True negatives
TOF	Time of flight
TP	True positives

Chapter 1 Introduction

1.1 Advanced Driver Assistance Systems (ADAS)

Modern vehicles are designed to attain an ever increasing level of drivability, manoeuvrability, safety and comfort. These very sought after attributes of vehicles are made possible with the aid of a large array of sensors and an unprecedented amount of computational power. Furthermore, the need to make vehicles aware of their surroundings has been at the heart of ADAS developments in recent years, to such an extent that even legislators are moving to accommodate vehicle and safety standards or impose new ones. ADAS innovations are shaping the automotive industry and the concept of autonomous driving has long been hailed as the way of the future. The driving experience alongside safety and efficiency are currently of utmost importance for car manufacturers in a highly competitive global market. Most existing or envisioned ADAS applications require scene perception which is attainable to varying degrees by leveraging the field of computer vision.

1.2 Motivation

ADAS applications are designed to either assist the driver in taking a decision, or in situations when a timely response is critical towards an appropriate outcome autonomously take action. Examples of such ADAS include pedestrian protection (Geronimo *et al.* 2010), lane departure warning (Son *et al.* 2015), collision avoidance and adaptive cruise control (Vahidi and Eskandarian 2003), traffic sign recognition and so on. These are all applications specific to urban-like on road environments but more recently the natural environment and off-road spectrum of ADAS started receiving attention particularly in vehicles that possess off-road capabilities. For example applications such as transparent bonnet could help a driver see the terrain during a steep ascent and manage steering action accordingly. In more common off-road situations terrain recognition technology (Gheorghe *et al.* 2015), (Tang and Breckon 2011) could help anticipate an appropriate terrain response. The latter ADAS example is the main driving factor behind the work presented in this thesis. Vehicle subsystems responsible for throttle response, suspension stiffness, direction response, differential locking and so on can be adjusted subject to driving conditions incurred by various terrain types. Configuring such vehicle subsystems to negotiate a certain terrain type leads to improved vehicle stability, drivability and manoeuvrability. The dynamic behaviour of a vehicle can be modelled given some known terrain adhesion and compressibility characteristics and a simple such example dealing with vehicle dynamics in vegetated terrain is provided by Talukder *et al.* (2002).

1.3 Project aim

Terrain perception is identified to be a prerequisite for any ADAS that would manage the terrain response of a vehicle. The main aim of the project is to produce an effective terrain recognition technology that can be accommodated later into an ADAS designed to improve the driving experience no matter the driving environment. At the moment a remote sensing solution,

in addition to normal tire pressure sensing and suspension readings, is needed to improve and anticipate an appropriate terrain response. This would complement the existing sensing on contact efforts based solely on CAN (controller area network) bus data (Taylor *et al.* 2012). As pointed by Khan (2013) terrain identification can be retrospective or prospective. Sensing on contact techniques estimate terrain retrospectively during vehicle traversal but a terrain response anticipating ADAS requires prospective terrain identification.

1.4 Research scope and objectives

To bring about a remote sensing solution that will address terrain recognition (i.e. prospective terrain identification) one needs to consider how the environment information can be acquired and what are the inherent strengths and weaknesses associated with a particular sensor. Generally, for all ADAS applications a number of sensors are currently available such as active ranging sensors including LIDAR (Badino, Huber and Kanade 2011), time of flight camera (TOF) (Zhu *et al.* 2008), radar and sonar as well as passive ranging with stereo (Einecke and Eggert 2014) and monocular vision with monocular camera (Alvarez *et al.* 2012a). The latter is often used not only to acquire 2D imagery but also to recover 3D information using 2D image sequences, a technique otherwise known as structure from motion (Sturgess *et al.* 2009). In addition to a sensor of choice one needs to look into possible ways of getting the vehicle to make sense of its sensory inputs and the popular fields of image processing and artificial intelligence (i.e. computer vision) have been explored in search of suitable data representation as well as hypothesis candidates, namely functions that will make predictions given the data representation. Loosely the data representation, commonly known as feature representation, constitutes the lens through which a learning algorithm of choice experiences the driving scene of a vehicle. Therefore in devising a terrain recognition strategy one needs to consider both what makes good terrain saliency as well as how it should be learned given the amount of ob-

tainable training data. To this end, semantic segmentation is the single most overlapping research field that can address the recognition task at hand. Generally semantic segmentation is comprised of partitioning the domain into atomic units (e.g. pixels, voxels, grid, superpixels and supervoxels) followed by category label assignment to every unit. Object detection is the alternative to semantic segmentation and both are able to solve a given recognition problem. By contrast, object detection yields localised classification by means of some template matching technique based on a sliding window. The end result represents a bounding box over a region of interest where template is maximally correlated. A major downfall of template matching is that bounding boxes do not generally span the objects of interest tightly. Performance of this already coarse classification technique becomes worse when classes are amorphous, as is the case with terrain classes. Semantic segmentation is therefore a natural choice with rich literature and existing methods expressive enough to deal with uncertain quantities and approximate reasoning. Several major research objectives are at the core of this thesis:

- Cast the prospective terrain recognition problem into a mid-level computer vision task, namely semantic segmentation of images recorded from a forward driving perspective.
- Devise a strategy to capture the saliency of terrain classes, in pursuit of environment perception (i.e. semantics of possibly generic driving scenes, including both on-road and off-road situations) using a monocular colour camera
- Semantically segment images recorded from a forward driving perspective into classes such as {grass, trees, sky, dirt, gravel, shrubs, tarmac and void}. Recall that terrain sensing is appealing from a vehicle drivability and manoeuvrability perspective. The terrain classes of choice contain multiple sub-classes themselves, however coarsely classified they could amount for a similar terrain response.

- Devise a strategy to improve upon the semantic segmentation by capturing and incorporating 3D real world terrain saliency obtainable with a ranging sensor. For example, stereo vision attainable using stereo camera is a promising technology and an active research topic (Scharstein and Szeliski 2002).
- Find suitable ways to incorporate prior knowledge about the problem domain into model predictions. This can potentially enable more refined semantic segmentations of terrain types present in both off-road and on-road driving scenes.

1.5 Thesis outline

This thesis is comprised of seven chapters and the entire logic flow behind it can be summarised using a dependency graph with each chapter represented by a node (Figure 1.1). The current chapter introduced the reader to the world of ADAS and defined the problem at hand while the remaining chapters are being outlined sequentially:

- Chapter 2

  Sets out to explore the research scope by reviewing the literature around semantic segmentation, predominantly considering works in the field of computer vision that achieve terrain and road recognition either explicitly or implicitly. Given that semantic segmentation is a widespread recognition task, comprised of techniques (e.g. feature extraction and classification framework) that can be ported from one application domain to another, some works are reviewed on the basis of being incident to the established research objectives. For example, with the advent of RGBD sensors semantic segmentation of indoor point clouds has been researched intensively leading to novel methods of representing geometric surface saliency or the interactions among 3D atomic units (i.e. segments). These concepts are in turn generic and therefore applicable to point clouds of both indoor and outdoor environments. Other recent works

dealing with semantic segmentation of images and considering irrelevant label sets but relevant techniques are also reviewed here. A table that puts the research scope into perspective alongside the developments outlined in Chapters 4, 5 and 6 respectively, is created as part of the literature review to enable the reader a localised understanding of the endeavours presented in this thesis.

- Chapter 3

  Introduces the popular discriminative learning algorithms that are prevalent in subsequent chapters. These algorithms have been used extensively for semantic segmentation in the literature and can be categorised depending on the type of predictions they make, namely scalar or structured. A scalar predicting algorithm outputs a single class label at each prediction step, effectively labelling a single segment (e.g. a pixel, voxel, grid cell, superpixel or supervoxel). Conversely, a structured predicting algorithm outputs a vector of labels, corresponding to a collection of segments that resemble a structured object, with each prediction step. The scalar predicting algorithms introduced are support vector machine (SVM), random forest (RF) and logistic regression (LR). The conditional random field (CRF) is introduced for structured output learning. Subsequent chapters show incremental developments with every chapter building on top of the previous one.

- Chapter 4

  Introduces the atomic units (i.e. segments) and dataset that are going to be used for semantic segmentation and sets out to find ways to capture the saliency of these atomic units or features that provide good discrimination among terrain classes using a monocular colour camera. In the absence of any depth information about the scene and knowing for a fact that plain colour appearance does not discriminate between some terrain classes such as dirt and gravel, texture is identified as a key saliency com-

ponent. Both scalar and structured output learning algorithms are explored in conjunction with several features of different texture flavours in a bid to devise an all-round model that is capable to make accurate terrain predictions across a driving scene image.

- Chapter 5

  Builds on top of the previous Chapter 4 by fixing the best texture alternative as a terrain saliency component and devises a strategy to incorporate an additional real world component obtainable with a ranging sensor of choice. Intuitively, terrain surface is as a discriminative property particularly able to set apart flat and bumpy segments even if all else fails (i.e. texture and colour). Two flavours of surface saliency are devised and explored alongside scalar and structured output learning algorithms in order to establish if there is anything to gain from a ranging sensor towards terrain recognition and how to leverage that. In the absence of annotated external datasets with reasonable variability among terrain classes, a public evaluation dataset for urban road terrain has been chosen to demonstrate tarmac recognition more objectively and mitigate the reliance on a proprietary data set. As one of the possible terrain classes, incurring a very distinctive terrain response due to virtually no compressibility and good surface adhesion, tarmac is in fact the most likely terrain type experienced by driving automobiles. Both flavours of surface saliency cast within a structured learning framework have been benchmarked. It is important to consider at this point that, is spite of the reasonable performance reported, labelling the tarmac superclass of urban road terrains is inherently detrimental to the road evaluation criteria. The road terrain recognition problem is more constrained than tarmac recognition and can be regarded as a subproblem. For example, evaluating the latter should not penalise labelling both roads and sidewalks as tarmac.

- Chapter 6

  Improves the semantic segmentation accuracy of urban road terrain reported in Chapter 5 by constraining the two devised models to not only consider smooth predictions as prior knowledge, but also road shape. This stems from the fact that urban roads are designed for vehicles with Ackermann steering. Vehicle's degrees of freedom in urban environments are restricted to physically and legally drivable areas. The terrain material of choice for such areas is typically tarmac, thus accurately solving the road recognition sub-problem provides for terrain perception and potentially for higher level reasoning about the scene too.

- Chapter 7

  This chapter concludes the thesis by succinctly discussing the contributions highlighted in Chapters 4, 5 and 6 and providing directions for future work.

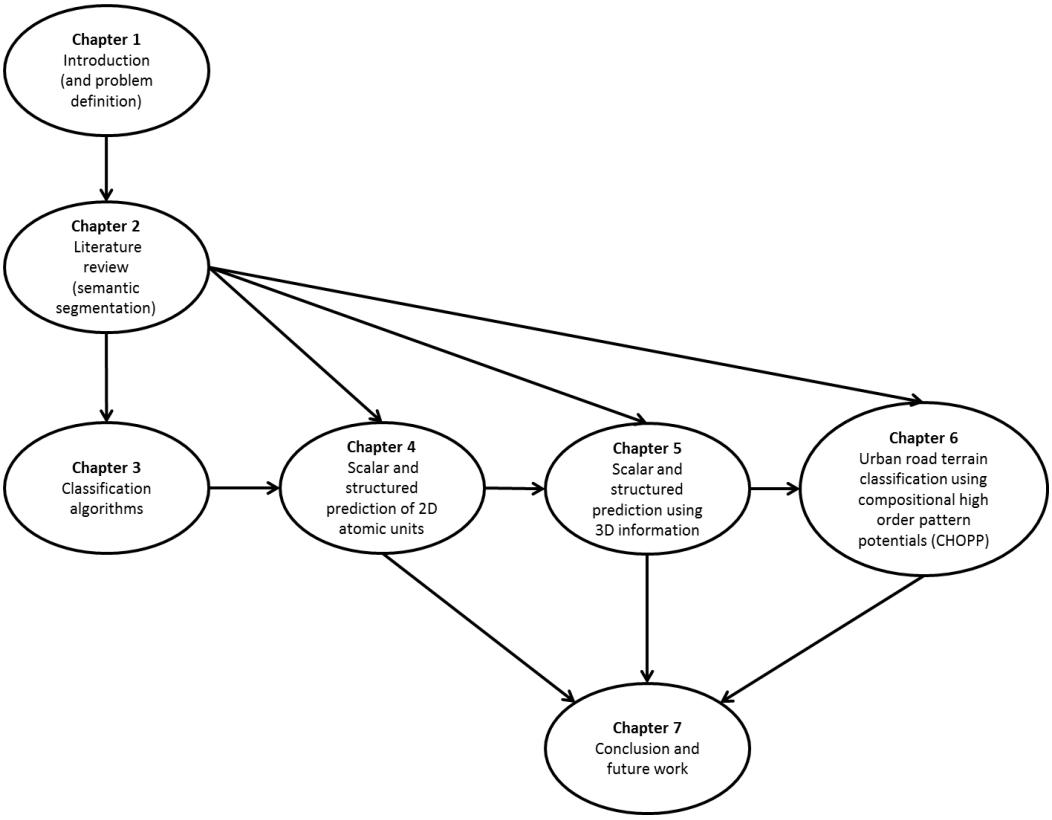


Figure 1.1 Thesis logic flow. Arrows represent dependencies among individual chapters

1.6 Contributions

Most notable contributions of this thesis can be collected from Chapters 4, 5 and 6 respectively. They will be enumerated here according to their significance, with first being the most significant contribution and subsequent ones with decaying levels of importance.

- Modelling the shape priors of urban roads with compositional high order pattern potentials. The contribution becomes particularly obvious by inspecting the literature review (Table 2.2) and localising Chapter 6. This corresponds to the research objective of finding suitable ways to refine predictions by exploiting prior knowledge about the problem domain.
- Exploring scalar output learning algorithms such as SVM, RF and LR alongside a structured output learning algorithm (i.e. CRF) with various flavours of texture in order to establish how to make good use of monocular vision for semantic segmentation of ter-

rains. Texture is the terrain saliency of choice due to ambiguous terrain colours. The structured output learning algorithm promotes smooth labelling by enforcing the prior knowledge that neighbouring segments of similar saliency are likely part of the same class. Fusing information from a ranging sensor demands additional computations and therefore it is important to establish how to leverage colour and texture first. With a fixed texture representation, the same selection of algorithms is used to establish if stereo vision can indeed improve classification performance across terrain classes. Experimental evidence suggests that stereo vision is not automatically superior to monocular vision. In fact among the selected algorithms, structured output learning has been the only technique able to boost the accuracy using stereo vision.

- Capturing the 3D surface saliency (or surface signature) of terrains from point clouds of both natural and urban outdoor environments using bag of spatial words with prototypes of height and fast point feature histograms. This is synonymous to feature extraction and produces a characteristic surface pattern (or signature) for every superpixel reconstructed as a supervoxel with stereo vision.
- Capturing the texture saliency of terrains from monocular vision using bag-of-visual-words with a high number of daisy descriptor prototypes, after experimenting with several texture descriptors including a lower quantisation level for daisy.
- Experimental evaluation of proposed learning algorithms and terrain saliency on the proprietary JLR dataset and the public KITTI dataset
- Extending the use of CHOPP augmented CRF to incorporate surface signatures from 3D atomic units such as supervoxels, in addition to colour and texture. This concept would be highly useful for other application domains and in particular for facial recognition where face shapes tend to be more constrained. In this context, shape and visual smoothness have already been considered jointly in the literature. However, a certain

spatial or geometric smoothness is to be expected since neighbouring atomic units situated on a face are likely to exhibit similar convexity.

1.6.1 Published work

The precursor to developments presented in Chapter 4, namely semantic segmentation albeit with a scalar predictor under different texture saliencies (Gheorghe *et al.* 2015), has been published in the international conference on systems engineering (ICSEng).

- Gheorghe, I., Li, W., Popham, T., and Burnham, K. J. (2015) 'Superpixel Based Semantic Segmentation for Assistance in Varying Terrain Driving Conditions'. *Progress in Systems Engineering*. Springer, 691-698
- Gheorghe, I., Li, W., Popham, T., Gaszczak, A., and Burnham, K. J. (2014) 'Key Learning Features as Means for Terrain Classification'. *Advances in Systems Science*. Springer, 273-282
- Haas, O., Kamran, S., Jaworski, P., and Gheorghe, I. (2013) 'Urban Traffic Simulators for Intelligent Transportation Systems'. *Measurement and Control* 46 (10), 309-314

Chapter 2 Literature review

2.1 Semantic segmentation and terrain recognition

Semantic segmentation is a very wide and active research area that is being pursuit in a domain specific fashion. Recent advancements in semantic segmentation have been driven by different application scopes, such ADAS for automotive, medical imaging, internet services (i.e. face tagging; image retrieval based on image semantics rather than text tags etc.), augmented reality and gesture control to name a few. Even though advancements in these fields are application oriented, having semantic segmentation as common denominator leaves room for more convergence between innovations. As the name suggests it, semantic segmentation refers to the process of assigning a meaning to every element, or atomic unit of a particular environment, acquired with the aid of sensing technology. In spite of the numerous endeavours only two types of publications will be reviewed here, namely those that set out to achieve similar labelling (e.g. urban or natural terrain classification) or those that present relevant techniques (e.g. feature extraction, classification framework) not necessarily in conjunction with an overlapping label set. In this work some recent advancements in semantic segmentation of indoor point clouds have been bridged together with recent developments in semantic labelling of general images in order to achieve state of the art outdoor terrain labelling (both on road and off-road) for ADAS development. Naturally, the relevant publications will be reviewed and put into perspective alongside with the contributions of this thesis. As far as the sensing technology goes, two types of atomic units are prevalent here, namely 2D segments obtainable via a

monocular camera and 3D segments obtainable via both passive and active ranging sensors. Passive ranging sensors include stereo cameras while active ranging sensors include LIDAR and RGBD sensors. Guiding the literature review in the direction of ADAS is not needed as it will become obvious that completely different ADAS applications have overlapping labelling needs and hence can be served by similar architectures. Indeed, much work has been done recently in classifying the road subclass of tarmac, especially for urban scenarios where roads are nothing but tarmac areas confined to a physically and legally drivable region. The push for a reliable solution in such cases is heavily motivated by ADAS where road occupancy is of paramount importance. For example, pedestrian protection (Geronimo *et al.* 2010), lane departure warning systems (Son *et al.* 2015), collision avoidance (Vahidi and Eskandarian 2003) and so on. Knowing where the road is helps towards higher level reasoning about a traffic scene (e.g. possible interactions) since the road spans the physical location of objects (Ess *et al.* 2009), (Geiger *et al.* 2014). It is obvious that such labelling of urban environments can indirectly provide for a totally different ADAS application where surface adhesion (or road grip) and terrain compressibility demand an appropriate vehicle behaviour. The labelling space of the latter ADAS would accept classification of both {road} and {sidewalk} classes as a {tarmac} superclass since they are generally made of the same material and therefore have the same adhesion and compressibility properties. Even though semantic segmentation for a road occupancy ADAS is more constrained, it would be sufficient for a road material ADAS since in urban environments vehicle's degrees of freedom are governed by regulations (e.g. sidewalks are usually designed for pedestrian use only). On the other hand driving in natural environments (i.e. off-road) requires a more relaxed material classification approach as such environments lack structure. In addition to that, the only factors restricting vehicle's degrees of freedom are large non-drivable obstacles (e.g. trees) (Manduchi *et al.* 2005). Other than categorising the set of labels and the type of segments they are attributed to, it is also important to review how prior knowledge (if any)

is incorporated into prediction of labels. Prior knowledge represents the information that is known beforehand about the driving scene. In a Bayesian sense, a conclusion or posterior can be obtained by combining the initial beliefs with new evidence when available. Road priors can be classified given the type of knowledge they represent. In the context of general semantic segmentation the following priors become apparent: location, temporal, smoothness and more recently global shape (Kae *et al.* 2013), (Yujia, Tarlow and Zemel 2013). Naturally, some of these priors are also encountered in the literature dealing with semantic segmentation of road or terrain scenes. Typically referred to as assumptions, they can be either learned by exploring a training set or simply imposed. For example, road location in an image can be learned in a supervised way by using ground truth annotations of training examples (Alvarez *et al.* 2013) or imposed as a hard assumption such as the image bottom (Alvarez *et al.* 2012a), below the horizon line, towards the vanishing point (Alvarez, Gevers and Lopez 2010) etc. Temporal coherence of image sequences is usually motivated by practical applications whereby the scene in front of the vehicle is likely to be similar from one frame to another unless severe steering is applied (Alvarez *et al.* 2013), (Alvarez, Gevers and Lopez 2010), (Beucher and Yu 1994). Finally, smoothness is usually encouraged via a probabilistic graphical model framework that makes joint predictions across the entire image effectively labelling all the individual pixels in a structured fashion (Alvarez *et al.* 2012b). Alternatively, this can be performed by over-segmenting the image into a grid or superpixels in order to reduce graph complexity and mitigate redundancies (Farabet *et al.* 2013). Working with atomic representation of images such as pixels or superpixels and predicting them individually means that even if a certain prior is advertised as a global shape prior (i.e. straight or turning road shape (Alvarez *et al.* 2013)), it is merely a location prior that helps to refine the prediction of a scalar label. An exceptional case would be if these atomic representations span the entire classifiable objects without exceeding their boundaries but this cannot be guaranteed and it is the job of a structured classifier to

learn how various atomic parts puzzle together to form objects and hence their global shape. Otherwise, shape priors are made explicit in the context of a structured predication framework that reasons about all the atomic units jointly (Kae *et al.* 2013), (Yujia, Tarlow and Zemel 2013). Whilst these priors can in theory be introduced selectively or combined at various levels within the classification architecture, current relevant road scene literature (Alvarez *et al.* 2013), (Alvarez, Gevers and Lopez 2010) makes use of location and temporal coherence priors as a top down refinement step applied in conjunction with the bottom up classification (scalar prediction) of pixels, grid or superpixels based on salient features. Furthermore these priors are derived from segments of perspective images. Similarly, such priors are encountered to varying extents in the literature that considers 3D atomic units such as voxels or supervoxels for classification. For example in the works of Douillard, Brooks and Ramos (2009), Lim and Suter (2009), Niemeyer, Rottensteiner and Soergel (2012) a smoothness prior is imposed between 3D segments essentially constraining neighbouring segments with similar saliency to have the same label. Voxels represent collections of many neighbouring 3D points, just as pixels represent collections of many neighbouring 2D perspective points. In theory, one pixel is a discrete representation within a digital image that corresponds to an infinite number of light intensities within a continuous image. A similar analogy can be made between voxels and real world continuous surface points. Moreover, just as pixels can be grouped together to form superpixels based on their location, texture and colour coherence (Achanta *et al.* 2012), voxels can also be grouped together (Douillard *et al.* 2011b), (Papon *et al.* 2013) in order to reduce redundancies, computation and the complexity of subsequent classification steps. The 3D grouping criteria are location and voxel proximity information or neighbouring distribution. Finding proximity information is often formulated as a plane fitting or normal estimation problem. In addition to that, colour information may also be used to guide voxel grouping and hence generate more accurate supervoxels. While urban environments permit the integration of more prior

knowledge to refine classifier predictions of a semantic segmentation task, natural environments are highly variable and unstructured allowing for only temporal or smoothness priors at most. There is no location or global shape that can be enforced upon natural environments. Even the location of a class such as {sky} can be different depending on vehicle tilt and therefore it is best avoided. In fact, the majority of publications that genuinely consider semantic segmentation of natural environments use a no prior, classifier only strategy (Angelova *et al.* 2007), (Bajracharya *et al.* 2008), (Filitchkin and Byl 2012), (Hadsell *et al.* 2009), (Jansen *et al.* 2005), (Lalonde *et al.* 2006), (Manduchi *et al.* 2005). In the next subsections more details will be provided about the literature considering the urban environment, the natural environment as well as the other work incident to this thesis either via feature extraction or classification framework.

Table 2.1 Literature review of semantic segmentation

semantic segmentation authors	label predictions								2D atomic units			3D atomic units									classes										
									camera			stereo			LIDAR			RGBD			driving scene										
	scalar			structured																											
	no prior/classifier only	location prior	temporal coherence	pairwise potentials		high order potentials		temporal coherence	pixels	grid	superpixels	voxels	grid	supervoxels	voxels	grid	supervoxels	voxels	grid	supervoxels	grass	tree	sky	dirt	gravel	shrubs	tarmac superclass	road subclass	void/other relevant	other irrelevant
				smoothness prior (sparse connectivity)	smoothness prior (dense connectivity)	smoothness prior	global shape prior																								
(Aijazi, Checchin and Trassoudaine 2013)	X																X					X						X	X	
(Alvarez *et al.* 2013)		X	X						X																			X	X	
(Alvarez *et al.* 2012a)		X							X																			X	X	
(Alvarez, Gevers and Lopez 2010)		X	X						X																			X	X	
(Alvarez *et al.* 2012b)				X					X													X	X					X	X	
(Alvarez, Salzmann and Barnes 2014)		X									X																	X	X	
(Alvarez, Salzmann and Barnes 2013)	X								X																			X	X	
(Anand *et al.* 2012)				X	X															X										X
(Angelova *et al.* 2007)	X										X										X			X	X		X		X	
(Arbeiter *et al.* 2012)	X																	X												X
(Bajracharya *et al.* 2008)	X								X																				X	
(Beucher and Yu 1994)			X								X																	X	X	
(Brust *et al.* 2015)		X							X													X	X					X	X	
(Campbell, Subr and Kautz 2013)					X				X																					X
(Douillard *et al.* 2011a)				X				X	X						X						X	X				X			X	
(Douillard, Brooks and Ramos 2009)	X			X											X						X	X				X	X		X	
(Ess *et al.* 2009)	X									X											X	X	X					X	X	
(Farabet *et al.* 2013)				X					X		X										X	X	X	X	X	X	X		X	X
(Fernandez-Maloigne and Bonnet 1995)	X								X																			X	X	
(Filitchkin and Byl 2012)	X								X												X			X	X		X		X	
(Fulkerson, Vedaldi and Soatto 2009)				X							X																		X	X
(Hadsell *et al.* 2009)	X								X																				X	
(Ibrahim and El-Saban 2011)				X		X			X																				X	X
(Jansen *et al.* 2005)	X								X												X	X	X	X	X	X			X	

Table 2.2 Literature review of semantic segmentation (continued)

semantic segmentation authors	label predictions								2D atomic units			3D atomic units									classes										
									camera			stereo			LIDAR			RGBD			driving scene										
	scalar			structured																											
	no prior/classifier only	location prior	temporal coherence	pairwise potentials		high order potentials		temporal coherence	pixels	grid	superpixels	voxels	grid	supervoxels	voxels	grid	supervoxels	voxels	grid	supervoxels	grass	tree	sky	dirt	gravel	shrubs	tarmac superclass	road subclass	void/other relevant	other irrelevant
				smoothness prior (sparse connectivity)	smoothness prior (dense connectivity)	smoothness prior	global shape prior																								
(Kae *et al.* 2013)				X			X																							X
(Kahler and Reid 2013)				X																X										X
(Khan, Komma and Zell 2011)	X									X											X				X		X		X	
(Krahenbuhl and Koltun 2012)					X				X												X	X	X				X		X	X
(Lalonde *et al.* 2006)	X														X														X	
(Li and Sahbi 2011)				X		X				X											X	X	X						X	X
(Lim and Suter 2009)				X													X				X	X							X	
(Manduchi et al. 2005)	X								X						X						X	X		X	X	X			X	
(Niemeyer, Rotten-steiner and Soergel 2012)				X											X							X				X	X		X	
(Passani, Yebes and Bergasa 2014)				X					X																			X	X	
(Rusu *et al.* 2009)				X														X												X
(Sturgess *et al.* 2009)				X		X			X			X										X	X					X	X	
(Vitor *et al.* 2013)	X										X																	X	X	
(Vitor, Victorino and Ferreira 2014)	X										X																	X	X	
(Wang, Fremont and Rodriguez 2014)	X								X																			X	X	
(Yujia, Tarlow and Zemel 2013)				X			X		X																					X
(Zhang, Wang and Yang 2010)				X				X			X			X								X	X					X	X	
(Zhang and Chen 2012)					X				X												X	X	X				X		X	X
Chapter 4	X										X										X	X	X	X	X	X	X		X	
Chapter 4				X							X										X	X	X	X	X	X	X		X	
Chapter 5	X										X			X							X	X	X	X	X	X	X		X	
Chapter 5				X							X			X							X	X	X	X	X	X	X		X	
Chapter 5				X							X			X														X	X	
Chapter 6				X			X				X			X														X	X	

2.2 Urban environment

Road terrain classification in urban environments has been pursued and reported in a number of previous publications. Some works consider the semantic segmentation of 2D atomic units obtainable via a monocular camera, while other focus on 3D atomic units obtainable via either stereo vision or LIDAR. Out of those that classify 2D atomic units some take a scalar labelling strategy to predict the labels of units individually with no prior (Alvarez, Salzmann and Barnes 2013), (Ess *et al.* 2009), (Fernandez-Maloigne and Bonnet 1995), (Vitor *et al.* 2013), (Vitor, Victorino and Ferreira 2014), (Wang, Fremont and Rodriguez 2014), with location prior (Alvarez *et al.* 2012a), (Alvarez, Salzmann and Barnes 2014), (Brust *et al.* 2015), with temporal coherence (Beucher and Yu 1994) or both location and temporal coherence (Alvarez *et al.* 2013), (Alvarez, Gevers and Lopez 2010). Others take a structured labelling strategy to predict all 2D units jointly using a smoothness prior (Alvarez *et al.* 2012b), (Passani, Yebes and Bergasa 2014).

The work of Fernandez-Maloigne and Bonnet (1995) is an early example of road segmentation using texture information in conjunction with a neural network. More recently Alvarez, Salzmann and Barnes (2013) learn the contribution of various colour spaces in a bid to create a robust appearance representation of road pixels based on training examples. Classification is then performed by applying a threshold on the linear combination of these colour spaces. Ess *et al.* (2009) classify individual grid patches of an image (i.e. scalar output) into a number of classes using visual information and additional depth information if available. Such semantic segmentation is used as a meta-representation towards higher level reasoning. This includes estimating the road type {left, right turn, straight, junction etc.} or simply detecting the presence of crossings, pedestrians and cars in the overall image. Vitor *et al.* (2013) use a stereo disparity map as a pre-processing step to coarsely threshold the reference image into multiple planes (i.e. horizontal and vertical) representative of the free space, obstacles and other ob-

jects within the road scene. Simultaneously, a watershed transform segments such image into superpixels. With each superpixel being described by a combination of both colour statistics and plane occurrences, an artificial neural network is employed for classification of all the atomic units into road and non-road. Incremental work with respect to (Vitor *et al.* 2013) is presented in (Vitor, Victorino and Ferreira 2014). Again the road scene image is segmented into superpixels using the watershed transform. However, superpixel features are obtained using bag-of-visual-words (i.e. colour, texture and pixel location) and bag of disparity words. Subsequently each superpixel is classified independently using an ensemble classifier. Wang, Fremont and Rodriguez (2014) attain pixel-wise classification of drivable area in an image by intersecting two binary maps obtained via thresholds. One map is generated using the disparity image and the other using a distribution of seed pixels in the log-chromaticity space. Alternatively, such binary maps are converted into likelihood maps whereby an element-wise multiplication between the two would yield a joint confidence map of road pixels. Still in the scalar labelling strategy of 2D segments but this time with location prior, Alvarez *et al.* (2012a) use a convolutional neural network (CNN) to predict each pixel as sky, road and vertical structures. The CNN is trained on machine annotated training sets and the road is assumed on the bottom part of an image. Alvarez, Salzmann and Barnes (2014) propose a direct superpixel labelling by a traffic-aware road prior. Such prior is obtained using the output responses of typical object detectors in traffic images to rule out non-road locations. As an alternative, a non-parametric road prior is proposed, by exploiting the ground truth of similar superpixels within the training set. Brust *et al.* (2015) encode spatial information using the absolute position of a patch in the fully connected layers of a CNN, thus effectively learning a road location prior from training data in addition to visual features. Some authors consider only temporal coherence such as Beucher and Yu (1994) where a watershed segmentation is applied to the gradient of a dynamic time filtered image resulting in crude superpixels which are classified based on their geo-

metric characteristics. Other authors consider the integration of both location and temporal coherence. For example Alvarez *et al.* (2013) attain pixel predictions (i.e. labelling) by placing a threshold on a confidence map obtained using a combination of location prior and illumination invariant statistics describing the road. A classifier is employed to learn and predict the location prior which is discretized into straight road, strong turns and soft turns. Prototypical road confidence shapes are learned from the training data by averaging the annotated ground truth within each category. Furthermore, the location prior estimation is subject to temporal coherence since perspective image sequences of a typical driving scenario contain smooth transitions between road shapes from one frame to another. Alvarez, Gevers and Lopez (2010) use a Bayesian framework to combine different cues extracted at the image level under the assumption that road is below the horizon line, aimed at the vanishing point and there is an inherent perspective layout and geometry of the road scene. This geometrical road prior is obtained by simply averaging the ground truth across the training samples. Temporal smoothing is applied to refine current predictions based on past ones. Structured prediction for 2D atomic units has also been explored before, effectively imposing a smoothness prior on the final labelling of a scene. Alvarez *et al.* (2012b) train a CNN to provide unary potentials for each class within a conditional random field framework (CRF), used for labelling consistency. Here classes that are typically represented by just a few pixels in an image {i.e. other} such as traffic sign, column/pole are filtered out. Both {sidewalks and road} are part of the tarmac superclass in a typical urban scenario, but they are assigned distinct labels. This again shows that semantic labelling is achieved for tarmac road, albeit from the perspective of a different category of ADAS, namely one that prioritises scene occupancy over the surface adhesion or compressibility. Similarly, Passani, Yebes and Bergasa (2014) apply a pairwise CRF on pixel atomic units but with hand-crafted features and on down-sampled images for faster processing.

Finally, semantic segmentation of urban environment has also been attempted with 3D atomic units from either stereo vision (Sturgess *et al.* 2009), (Zhang, Wang and Yang 2010) or LIDAR (Aijazi, Checchin and Trassoudaine 2013), (Douillard *et al.* 2011a), (Douillard, Brooks and Ramos 2009), (Lim and Suter 2009), (Niemeyer, Rottensteiner and Soergel 2012). Out of these works only Aijazi, Checchin and Trassoudaine (2013) perform scalar label predictions with no prior. Here the points acquired via a LIDAR are turned into voxels and further augmented with reflectance intensity, RGB colour as well as surface normals in a bid to segment the scene into coherent supervoxels. After segmentation, a semantic label is attributed to each supervoxel individually based on predefined thresholds of surface normals, height, colour and geometrical shape. The rest of works perform structured label predictions with smoothness prior. Among these, some consider the integration of temporal coherence given the sequential nature of scenes such as Zhang, Wang and Yang (2010). Douillard *et al.* (2011a) project the returns of 2D laser scans in the image domain in order to seed regions of interest (ROI). The ROI's size adapts with the distance of a point in the scan and ROI's features encode both visual appearance as well as the corresponding geometric properties of the laser map. Given the nature of the laser scans, a chain CRF framework is proposed to allow for structured prediction. This is extended to a more general CRF (i.e. lattice like) by considering temporal links between scans, effectively disguising temporal coherence into a smoothness prior. The label of each ROI is aggregated at the seed locations of the scan projections within an image. The major limitation of this work is that, in spite of temporal linking, a lot contextual information is lost when considering only a 2D semantic map. A fixed pitch angle of the laser scanner makes it impossible to consider classes if they are below a certain height. However the graphical structure allows for interchangeability between temporal and spatial linking which renders the method usable with full 3D scans via a LIDAR. Douillard, Brooks and Ramos (2009) use the CRF framework sparingly to predict only labels for ground voxels as {grass, asphalt} in an urban scenario. More elevated

voxels are clustered into objects and classified scalarly in a different step using a discriminative classifier. Each voxel is augmented with visual appearance information coming from corresponding ROI within the perspective image. Lim and Suter (2009) segment LIDAR point clouds into supervoxels and represent them by their colour, reflectance and geometry features. Structured prediction of supervoxel semantics is achieved using a CRF with multi-scale pairwise connections. Niemeyer, Rottensteiner and Soergel (2012) generate point clouds using airborne LIDAR in urban areas and each voxel is classified into asphalt ground, natural ground, building, low vegetation and tree. Therefore the non-asphalt terrain classes are merged into a single natural ground class. Features include intensity, distance to ground and distribution saliency derived from local point covariance matrix. Structured prediction is achieved using a pairwise CRF where local node links are established via a nearest neighbour search in voxel space. Edge potentials are computed using absolute difference between the feature vectors of neighbouring voxels. In contrast, the works of Sturgess *et al.* (2009) and Zhang, Wang and Yang (2010) use stereo vision to extract structure from motion features for each atomic unit. Since these features correspond to projections of 3D atomic units of point clouds into 2D perspective units labelling can be attributed to both. More specifically Sturgess *et al.* (2009) perform pixel-wise semantic segmentation of an entire road scene. Crude structure from motion features are projected from 3D point clouds to 2D image plane and used in conjunction with appearance features. A CRF is augmented with high order smoothness potentials, essentially adding a cost to assigning different pixel labels within a superpixel segment. The quality of the superpixel segments is also accounted for in the formulation of such potentials. Zhang, Wang and Yang (2010) achieve semantic segmentation of urban scenes via structured prediction of superpixel labels using a Markov random field (MRF). Superpixel unary potentials are the classification scores of a random forest classifier that uses only 3D features from dense depth maps obtained via a video sequence. Superpixels are regarded as 3D patches and their saliency is cap-

tured using point's normal, height, local and neighbouring planarity as well as the distance to camera path. Pairwise potentials on the other hand are only a measure of colour similarity between neighbouring superpixels. In addition a pixel-wise temporal fusion is applied to smooth prediction by considering neighbouring frames.

2.3 Natural environment

Semantic segmentation of natural environments, including terrain type classification has been tackled in a number of previous specialised works using both 2D atomic units of monocular vision (Bajracharya *et al.* 2008), (Hadsell *et al.* 2009), (Jansen *et al.* 2005) and 3D atomic units of LIDAR (Lalonde *et al.* 2006), (Manduchi *et al.* 2005). Terrain classification has also been considered in a more generic sense with both natural and man-made terrains i.e. {tarmac} in (Angelova *et al.* 2007), (Filitchkin and Byl 2012), (Khan, Komma and Zell 2011) where the input sensor is again a monocular camera. A common trait among all these works appears to be the intentional omission of priors and the adoption of classifier only techniques. This is not surprising since unlike their urban counterpart, natural environments lack structure which makes classification harder (Manduchi *et al.* 2005). However, the general semantic segmentation literature touches down on some natural environment and terrain classes (e.g. {grass, tree, sky, gravel, road} among other irrelevant ones) with smoothness priors (Farabet *et al.* 2013), (Li and Sahbi 2011), (Zhang and Chen 2012). It is therefore still reasonable to assume that segments with similar saliency are likely segments of the same class. The works dealing with 2D atomic units will be reviewed in more detail followed by those dealing with 3D segments. The work of Angelova *et al.* (2007) uses an ensemble of classifiers to distinguish among several terrain classes within image patches. Feature representation becomes more complex only when classes are visually similar and hard to distinguish. Bajracharya *et al.* (2008) use colour information to classify the scene into traversable and non-traversable terrains. Stereo vision is

utilised only at close range as part of a pre-classification step in order to find seed image locations for feature extraction. The training set is updated every few meters and subsequently all individual pixels within an image are classified by a trained linear SVM. Filitchkin and Byl (2012) consider terrain classification for a robotic application, namely a legged robot with adjustable gait. A linear SVM is trained to distinguish between several terrain classes described via bag-of-words features based on pixel descriptors. Hadsell *et al.* (2009) use a stereo module as a supervisor that assigns labels at close range thus creating training examples. Subsequently, a classifier is trained at every frame using lagging examples in order to achieve fast adaptability. Features are learned offline by using a multi-layer CNN. Jansen *et al.* (2005) fit colour distributions in images with Gaussian mixture models (GMM) and pixel classification of terrain types is performed using maximum likelihood. It has been identified here that recognising materials in natural terrain leads to improved vehicle drivability. From an application perspective the research of Jansen *et al.* (2005) is maximally aligned with the work being presented in this thesis. Khan, Komma and Zell (2011) investigate terrain classification from a mobile platform with a down facing camera. A high resolution image is over-segmented into a grid where each individual cell is classified scalarly by a random forest classifier. In this context several texture descriptors are evaluated, including local binary patterns and key point detectors such as daisy (Tola, Lepetit and Fua 2010). Daisy and other descriptors are used directly as features and are computed at only one seed location per cell. Lastly, the natural environment has also been semantically segmented using 3D atomic units. Lalonde *et al.* (2006) classify natural terrain into three classes namely scatter, linear and surface using LIDAR points as atomic units. Saliency features based on covariance matrix in point neighbourhoods are fitted by Gaussian mixture models (GMM). Subsequently, classification of each voxel is performed using a Bayesian classifier. Manduchi *et al.* (2005) present two methods for terrain cover perception, one using colour and the other using LIDAR range. Pixel-wise classification is achieved by a maximum likeli-

hood classifier, where the class-conditional likelihood is learned from training samples using a GMM. As a separate solution, single axis LIDAR returns are classified by analysing statistical models of range in a bid to discriminate between scattered (e.g. grass) and smooth (e.g. soil) patterns. The specialised literature around terrain classification in natural environments is still scarce but some notable endeavours have their origins in the learning applied to ground vehicles (LAGR) program under the umbrella of defence advanced research projects agency (DARPA) (Angelova *et al.* 2007), (Bajracharya *et al.* 2008), (Hadsell *et al.* 2009). Terrain perception has so far been researched predominantly in the context of robotic applications including unmanned ground vehicles (UGV) in natural environments where terrain traversability is key to robot functionality (Angelova *et al.* 2007), (Bajracharya *et al.* 2008), (Filitchkin and Byl 2012), (Hadsell *et al.* 2009), (Manduchi *et al.* 2005).

Works in semantic segmentation of images depicting general scenes often consider a label set that is overlapping to certain terrain classes, however they consider many classes (some irrelevant too e.g. cow, plane) and a disproportionate amount of training examples (i.e. only few samples per class) and therefore lack the much needed intra-class variability as required for an ADAS application. For the sake of completeness, since they make use of smoothness priors in conjunction with terrain types, they shall be reviewed here too. Farabet *et al.* (2013) use a multi scale convolutional neural net to classify every pixel in an image. First the input image is transformed using a Laplacian pyramid and then copies of the same convolutional network learn features at each scale thereby not only capturing texture and shape but also more context. This has been shown to counterbalance the need to enforce label consistency and scene level relationships via structured learning. Li and Sahbi (2011) segment images into grid atomic units and label them using a CRF augmented with higher order neighbourhoods. Smoothness has been enforced by considering pairwise potentials at four neighbouring cells as well as higher order potentials based on cell groupings. Zhang and Chen (2012) use a fully connected

pairwise CRF by linking together the hidden variables of all pixel nodes. Dense graph connectivity is the alternative to higher order potentials that would enable a CRF to incorporate more context for better semantic segmentation essentially permitting long range interactions between variables of interest.

2.4 Relevant techniques

Other works in the field of semantic segmentation are appealing and incident to the work presented in this thesis in two ways: either because of how they encode interactions between atomic units, or because of how their saliency is captured. The former aspect refers to the classification framework, whilst the latter refers to feature extraction. Indeed, recent work of semantic segmentation for general scenes tackle the labelling problem of atomic units with structured prediction under the popular conditional random field (CRF) (Lafferty, McCallum and Pereira 2001) or structural support vector machine (SSVM) (Tsochantaridis *et al.* 2005). Such frameworks allow for all atomic units to be classified jointly whilst also considering how interactions among variables would influence labelling. For most part of this thesis, only developments of CRF will be reviewed and further utilised. The connectivity of such a graphical model has traditionally been restricted, allowing for each hidden variable representative of an atomic unit to only depend on a few neighbouring variables in order to make learning and inference efficient (Farabet *et al.* 2013), (Fulkerson, Vedaldi and Soatto 2009). Smoothness priors via pairwise connections gained popularity for semantic segmentation of images after the introduction of CRF by Lafferty, McCallum and Pereira (2001). In order to increase the quality of segmentation in such restricted connectivity models, some works have introduced higher order potential functions that would depend on more than two atomic units at a time being able to catch longer ranges of interactions (i.e. between more than the immediate neighbours). Such models often encourage consistency or smoothness in higher order neighbour-

hoods (Ibrahim and El-Saban 2011), (Li and Sahbi 2011). More recently, very promising results have been obtained with restricted connectivity models by augmenting the pairwise CRF with higher order compositional pattern potentials (Yujia, Tarlow and Zemel 2013), most notably a global shape prior learned using a restricted Boltzmann machine (RBM) (Kae *et al.* 2013), (Yujia, Tarlow and Zemel 2013). Such models are still in their infancy but are very appealing because they enforce both local labelling consistency as well as global shape, something that the traditional models would benefit from since they tend to oversmooth labelling around the edges of objects. Other than augmenting the CRF with higher order potentials there is also the option to use only pairwise potentials but move to densely connected models (i.e. link each node with the rest) and some works have done that by making certain assumptions such as having Gaussian edge potentials (Campbell, Subr and Kautz 2013), (Krahenbuhl and Koltun 2012) or spatial stationarity (Zhang and Chen 2012).

The recent emergence of commercially available (and cheap) RGBD sensors has opened possibilities for a number of new applications especially in indoor environments, such as manipulating multimedia systems with gesture recognition technology or robotic manipulation of semantic objects. The prospect of such applications has stirred up a lot of excitement and much research has gone into the semantic segmentation of a 3D indoor scene. Naturally some mature concepts from 2D semantic segmentation of images have been brought forward into the 3D realm. For example, relational reasoning about the atomic units under a probabilistic framework with models such as the CRF (Kahler and Reid 2013), (Rusu *et al.* 2009) or SSVM (Anand *et al.* 2012), feature representation with popular image descriptors (extended to 3D) and even vector quantisation techniques such as bag of visual (and the analogous depth) words (Hernandez-Vela *et al.* 2012). Intuitively, the role of structured prediction remains the same as with 2D segments. A smoothness prior for instance encourages similar segments to have the same label. The difference is that the measure of similarity now captures both visual

appearance as well as surfaceness such as the coplanarity and convexity between segments (Anand *et al.* 2012). Conversely, some characteristic novel features and novel point cloud descriptors have also been developed in a bid to better capture saliency of 3D atomic units. One notable such example is the fast point feature histogram (FPFH) (Rusu, Blodow and Beetz 2009), a point descriptor that captures surface saliency based on the point normals of a neighbourhood. The FPFH descriptor is independent of the view point and has been found to provide good discrimination between primitive geometric surfaces by Arbeiter *et al.* (2012) and particularly in conjunction with a structured learning framework by Rusu *et al.* (2009). Moreover, it does not consider the distance between points as a feature which makes it suitable for datasets where far away points are inherently further spaced from each other. The ramification of such studies will be propagated later in the thesis to achieve semantic segmentation in outdoor scenes both in natural and urban environments.

Chapter 3 Learning algorithms

3.1 Variables of interest

To begin with, this chapter introduces notations for variables of interest used throughout the thesis and shared by learning algorithms in their formalisms. With few distinct algorithms introduced, the mathematical notations used here follow those of Kae *et al.* (2013) very closely.

- Assume that an image I (or point cloud for that matter) is comprised of $S^{(I)}$ atomic units or segments, with $S^{(I)}$ not necessarily constant over different images or point clouds. In conjuction with algorithms that resemble probabilistic graphical models such as the logistic regression and the more general conditional random field, assume that these segments correspond to nodes in an undirected graph. Although most learning algorithms have a probabilistic interpretation, segments used elsewhere will merely correspond to data samples.

- Let $V^{(I)} = \{1, \dots, S^{(I)}\}$ denote the set of segment nodes of I.

- Let $E^{(I)} = \{(i,j) : i,j \in V^{(I)} \; and \; i,j \; adjacent \; segments\}$ denote the set of segment edges.

- Let $G^{(I)} = \{V^{(I)}, E^{(I)}\}$ be the undirected graph corresponding to I.

- Let $X_V^{(I)} = \{x_s^{node} \in \mathbb{R}^{D_n}, s \in V^{(I)}\}$ denote the set of node features corresponding to I.

- D_n denotes the dimension of the node features

- Let $X_E^{(I)} = \{x_{i,j}^{edge} \in \mathbb{R}^{D_e}, (i,j) \in E^{(I)}\}$ denote the set of edge features corresponding to I.

  - D_e denotes the dimension of the edge features

- Let $X^{(I)} = \{X_V^{(I)}, X_E^{(I)}\}$ denote the set of node and edge features extracted from I.

- Let $Y^{(I)} = \{y_s \in \{0,1\}^L, s \in V^{(I)}: \sum_{l=1}^{L} y_{sl} = 1\}$ denote the set labels corresponding to the nodes of I. Note that some algorithms might tailor their target values y_s differently to help formulate objective functions, but is should be clear from the context.

  - L denotes the number of labels and throughout this thesis is either 8 for predominantly natural environments (i.e. {grass, trees, sky, dirt, gravel, shrubs, tarmac and void}) or 2 for urban environments (i.e. {tarmac and void}).

The remainder of this chapter argues in favour of discriminative models. Moreover, it briefly shows how the predicting hypothesis specific to each algorithm is learned as well as how the actual predictions (i.e. inference and labelling) come about given the segment features.

3.2 Discriminative vs generative models

By simply inspecting Bayes probability rule it can be observed that labelling Y of the object we are trying to predict given some features X can be achieved in two ways. Either by directly modelling a conditional distribution of labels given the features $P(Y|X)$ or by modelling an intermediate joint distribution of labels and features $P(Y, X)$ from which $P(Y|X)$ can be eventually reasoned using Bayes formula. The former approach is called discriminative, while the latter approach is called generative. Assuming that such probability distributions are learned

given some amount of training data, a discriminative learning model is able to only sample hidden variables (i.e. labels) given the observed variables or features. On the other hand, a generative approach is able to sample (or generate) instances of any variable in the model.

$$P(Y|X) = \frac{P(Y,X)}{P(X)} \tag{3.1}$$

$$P(Y,X) = P(X|Y)P(Y) \tag{3.2}$$

While both models can be utilised for a given classification problem, a generative model would typically require modelling of $P(X|Y)$ as an intermediate step, that is the distribution across features given the labels. Assuming independence among such observations given the possible states will impact classification accuracy, particularly if features have unaccounted correlations (Sutton and McCallum 2006). In contrast, classification under a discriminative approach follows $P(Y|X)$ directly without a need for modelling $P(X|Y)$. In this work, a number of discriminative algorithms will be utilised to achieve semantic segmentation of superpixels from image frames. Discriminative algorithms (as well as the generative) can be categorized into scalar and structured prediction algorithms. Discriminative scalar prediction algorithms such as support vector machine (SVM), random forest (RF) and logistic regression (LR) will assign a label to a single superpixel at each prediction step. In contrast, with each discriminative structured prediction using the conditional random field (CRF) model, all segments within an image frame (or point cloud) will be classified.

3.3 Scalar prediction

Scalar predictors use only the set of node features X_V towards their predictions and for every labelling Y they do not take context information into consideration.

3.3.1 Support vector machine

Support vector machine (SVM) is a popular discriminative learning algorithm that has the ability to learn optimal separation between classes in a high dimensional space. Learning an optimal hyper plane parameterisation is cast as a constrained optimisation problem by considering the distance or proximity of training examples (support vectors) with respect to a hypothesis in addition to minimising the training error. Generally it draws on the benefit of using kernel tricks (essentially mapping feature vectors into higher dimensions) in order to separate nonlinearly separable classes. Distance is frequently defined in the Euclidean sense as the l_2 norm. Besides the myriad of works on the topic of SVM learning the reader is invited to explore the work of Chapelle, Haffner and Vapnik (1999) where the authors advocate the use of SVMs for classification of images represented using histograms. Motivated by this study, an SVM with radial basis function (RBF) kernel has been used for multi-label classification of superpixels in the one-vs.-one setting. This multi-label strategy constructs $\frac{L(L-1)}{2}$ classifiers each being fitted for a pair of classes. During testing, all classifiers vote for a class and the class receiving the majority of votes becomes the prediction. The SVM implementation of Pedregosa *et al.* (2011) has been used for experiments throughout this work.

3.3.1.1 Learning and prediction

For every pair of classes, learning requires their corresponding N training segments of the form $\{(x_s, y_s) : x_s \in \mathbb{R}^{D_n}, y_s \in \{-1,1\}\}_{s=1}^{N}$ from the original training set, where y_s is the class label of sample x_s. Note that for this binary classification the negative class is represented with -1 instead of 0. For nonlinear SVM, features x_i and x_j of segments indexed with i and j are never explicitly mapped into a high dimensional space since the required dot product of their mapping is more easily obtainable using a Mercer kernel $K(x_i, x_j)$. The Mercer kernel of choice is the Gaussian RBF and the SVM is trained by maximizing:

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{3.3}$$

$$K(x_i, x_j) = e^{\left(-\gamma \|x_i - x_j\|^2\right)}, \gamma > 0 \tag{3.4}$$

Here α is a vector of N non negative Lagrange multipliers corresponding to constraints and data samples with $\alpha_i > 0$ are called support vectors. To make a prediction with one of the $\frac{L(L-1)}{2}$ classifiers for a new (i.e. unseen) segment with feature ${x'}_s$ a hypothesis function $h({x'}_s)$ is evaluated:

$$h({x'}_s) = sgn\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, {x'}_s) + b\right) \tag{3.5}$$

The class receiving the majority of votes from classifier evaluations wins. To label an entire image I (or point cloud) this process must be repeated for all segments with ${x'}_s \in X_V^{(I)}$.

3.3.2 Random forest

Random forest (RF) classifier (Breiman 2001) is an ensemble learning method that grows a collection of tree classifiers, subsequently contributing towards class prediction via a majority voting procedure. Creating bootstrap samples from a training set (i.e. sampling with replacement) to grow each tree followed by plurality voting for classification is often called tree bagging. Departing from this, random forests use random feature selection to make splits at each node while growing the tree. The resulting classifier is faster than other ensemble learning methods such as plain tree bagging or boosting and robust to outliers and noise. Adding more trees does not result in overfitting, in fact it allows for an upper bound to be placed on the generalisation error at the expense of increased run-time complexity for the classification process. Motivated by the work of Khan, Komma and Zell (2011), a number of 100 forest estimators have been selected to label every segment unit (e.g. superpixel). The RF implementation of Pedregosa *et al.* (2011) has been used for experiments throughout this work.

3.3.2.1 *Learning and prediction*

Starting with a set of N training segments of the form $\{(x_s, y_s) : x_s \in \mathbb{R}^{D_n}, y_s \in \{1, \dots, L\}\}_{s=1}^{N}$ the objective is to train T classification trees that can predict a label for unseen samples x'_s with their corresponding hypotheses $\{h_1(x'_s), \dots, h_T(x'_s)\}$. Individual hypotheses are aggregated into a combined hypothesis $h_c(x'_s)$ in order to predict a segment label by plurality voting. The procedure for the learning phase of a random forest is as follows:

- For each classification tree $t \in \{1, \dots, T\}$ choose a random subset of $n < N$ training segments with replacement from the original training set.
- Grow the tree without pruning by randomly choosing a subset of $m < D_n$ features to make the best split at each node untill a maximum tree depth is attained.
- Return $\{h_1, \dots h_T\}$.

To label an entire image I (or point cloud) the hypothesis $h_c(x'_s)$ must be evaluated for all segments with $x'_s \in X_V^{(I)}$.

3.3.3 Logistic regression

In its simplest form, logistic regression (LR) is a scalar binary classifier that can be regarded as a special case of a more general conditional random field. In fact the segments of an entire image I (or point cloud) can be labelled using the energy function of a conditional random field by omitting the energy terms accounting for links (i.e. with edge features X_E) between the unobserved nodes. Features extracted for each segment are simply the node features X_V thus only the unary potentials of segments are used for labelling. The LR implementation of Kae *et al.* (2013) has been used for experiments throughout this work.

- For more intuition, consider the simple example where the most probable binary label y needs to be estimated for a segment x. In general if some $P(y = 1) = p$ and $P(y = 0) = 1 - p$, a Bernoulli distribution can be used to express this compactly:

$$P_{Bernoulli}(y) = p^y(1-p)^{1-y}, \qquad y \in \{0,1\} \tag{3.6}$$

$$P_{LR}(y = 1|x) = \frac{exp(\beta x)}{1+exp(\beta x)} = p \tag{3.7}$$

Where β denotes a vector of parameters that are learned using training data.

$$P_{LR}(y|x) = \left(\frac{exp(\beta x)}{1 + exp(\beta x)}\right)^y \left(1 - \frac{exp(\beta x)}{1 + exp(\beta x)}\right)^{1-y} = \frac{exp(y\beta x)}{1 + exp(\beta x)} \tag{3.8}$$

3.3.3.1 *Learning and prediction*

To label a collection of segments, the most probable labels must be selected given the node features and the learned parameters $\Gamma \in \mathbb{R}^{L \times D_n}$.

$$P_{LR}(Y|X) \propto exp\big(-Energy_{LR}(Y,X)\big) \tag{3.9}$$

$$Energy_{LR}(Y,X) = Energy_{node}(Y,X_V) \tag{3.10}$$

$$Eenergy_{node}(Y,X_V) = -\sum_{s \in V}\sum_{l=1}^{L}\sum_{d=1}^{D_n} y_{sl}\Gamma_{ld}x_{sd} \tag{3.11}$$

Note the similarities between the numerator of Equation (3.8) and Equation (3.11). Given the training data $\{Y^{(m)}, X^{(m)}\}_{m=1}^{M}$ comprised of M segmented images (or point clouds) parameters Γ are learned by maximising the conditional log likelihood L.

$$L = \max_{\Gamma} \sum_{m=1}^{M} \log P_{LR}(Y^{(m)}|X^{(m)}) \tag{3.12}$$

$$\frac{\partial L}{\partial \Gamma_{ld}} = \frac{1}{M}\sum_{m=1}^{M}\left(\sum_{s\in V^{(m)}} y_{sl}\, x_{sd} - \sum_{s\in V^{(m)}} P(y_{sl}|x)\, x_{sd}\right) \tag{3.13}$$

To estimate the posterior probability of labelling segment s with label l given the segment feature x_s:

$$f_{sl}^{node}(X_V, \Gamma) = \sum_{d} x_{sd}\Gamma_{dl} \tag{3.14}$$

$$P_{LR}(y_s = l|x_s) = \frac{exp\left(f_{sl}^{node}\right)}{\sum_{l'} exp\left(f_{sl}^{node}\right)} \tag{3.15}$$

3.4 Structured prediction

Structured predictors use both the set of node features X_V and the set of edge features X_E towards their predictions, taking context information into consideration for every labelling Y. Frequently encountered algorithms in the semantic segmentation literature are the conditional random field (CRF) (Lafferty, McCallum and Pereira 2001) and the structural support vector machine (SSVM) (Tsochantaridis *et al.* 2005). Details of the former will be provided here.

3.4.1 Conditional random field

Conditional random field (Lafferty, McCallum and Pereira 2001), (Sutton and McCallum 2006) is particularly useful for structured prediction problems and has been used intensively for semantic segmentation of images (Krahenbuhl and Koltun 2012), (Li and Sahbi 2011), (Sturgess *et al.* 2009). It models a conditional joint distribution by accounting for the relationship between neighbouring segments represented by a Markov network on graph $G(V^{(I)}, E^{(I)})$. Here V includes the unobserved and observed nodes within the random field of I and E is the set of edges (i, j) relating adjacent segments. X is comprised of both node features X_V and edge features X_E. Edge potentials are typically formulated to promote local smoothness among neighbouring segments. A label transition between segments that share a boundary takes into

account how similar and conversely how dissimilar their saliency is by means of a penalty term. The CRF implementation of Kae *et al.* (2013) has been used for experiments throughout this work.

3.4.1.1 Learning and prediction

To label a collection of segments, the most probable labels must be selected given the node and edge features as well as the learned parameters Γ and Ψ .

$$P_{CRF}(Y|X) \propto exp\big(-Energy_{CRF}(Y,X)\big) \tag{3.16}$$

$$Energy_{CRF}(Y,X) = Energy_{node}(Y,X_V) + Energy_{edge}(Y,X_E) \tag{3.17}$$

$$Energy_{node}(Y,X_V) = -\sum_{s\in V}\sum_{l=1}^{L}\sum_{d=1}^{D_n} y_{sl}\Gamma_{ld}x_{sd} \tag{3.18}$$

$$Energy_{edge}(Y,X_E) = -\sum_{(i,j)\in E}\sum_{l,l'=1}^{L}\sum_{e=1}^{D_e} y_{il}y_{jl'}\Psi_{ll'e}x_{ije} \tag{3.19}$$

In this formulation, $\Gamma \in \mathbb{R}^{L\times D_n}$ are the node weights and $\Psi \in \mathbb{R}^{L\times L\times D_e}$ are the edge weights, with L being the number of labels, D_n and D_e the dimensions of the node and edge features. Defining node and edge potentials for larger segments (e.g. superpixels in an image or supervoxels in a point cloud) rather than smaller atomic units (e.g. pixels or voxels) allows for a simpler graph and makes the approximate inference more efficient. Edges are connecting only adjacent (i.e. neighbouring) segments within I. Because loops may now be present in the graph, only approximate inference is possible and there are several techniques typically used. These include variational approaches, loopy belief propagation and Markov-chain Monte Carlo (MCMC). As pointed by Kae *et al.* (2013) for this inference scenario, a variational approach (i.e. mean-field inference) (Saul, Jaakkola and Jordan 1996) is suitable since it is guaranteed to converge usually to some local optimum, unlike loopy belief propagation. Moreover, it is faster than MCMC approaches albeit less accurate. Parameter learning given some training data

$\{Y^{(m)}, X^{(m)}\}_{m=1}^{M}$ consisting of M segmented images (or point clouds) is performed by maximising the conditional log likelihood (Kae *et al.* 2013):

$$L = \max_{\Gamma,\Psi} \sum_{m=1}^{M} \log P_{CRF}(Y^{(m)}|X^{(m)}) \tag{3.20}$$

$$\frac{\partial L}{\partial \Gamma_{ld}} = \frac{1}{M} \sum_{m=1}^{M} \left(\sum_{s \in V^{(m)}} y_{sl}\, x_{sd} - \sum_{s \in V^{(m)}} P(y_{sl}|x)\, x_{sd} \right) \tag{3.21}$$

$$\frac{\partial L}{\partial \Psi_{ll'e}} = \frac{1}{M} \sum_{m=1}^{M} \left(\sum_{(i,j) \in E^{(m)}} y_{il} y_{jl'} x_{ije} - \sum_{(i,j) \in E^{(m)}} P(y_{il}|x) P\big(y_{jl}|x\big) x_{ije} \right) \tag{3.22}$$

Using a variational approach $P_{CRF}(Y|X)$ is approximated by a simpler graphical model $Q(Y;\mu)$ which is parameterized by minimising the Kullback-Leibler divergence (KL) between the two probability distributions:

$$KL\big(Q(Y;\mu)||P_{CRF}(Y|X)\big) \tag{3.23}$$

For the purpose of mean field inference it is assumed that nodes of the approximating graphical model $Q(Y;\mu)$ are independent:

$$Q(Y;\mu) = \prod_{s \in V} Q(y_s) \tag{3.24}$$

$$Q(y_s = l) = \mu_{sl} \tag{3.25}$$

Parameters $\mu_{sl}^{(i)}$ are basically the posterior probability estimates of labelling segment s with label l and get updated over a number of iterations (i.e. until convergence or predefined) indexed by i. To start with $\mu_{sl}^{(0)}$ is initialized with the logistic regression estimates using only the node energies f_{sl}^{node}, followed by subsequent updates using both node and edge energies f_{sl}^{edge}.

$$f_{sl}^{node}(X_V, \Gamma) = \sum_{d} x_{sd} \Gamma_{dl} \tag{3.26}$$

$$f_{sl}^{edge}(\mu; X_E, E, \Psi) = \sum_{j:(s,j)\in E} \sum_{l',e} \mu_{jl'} \Psi_{ll'e} x_{sje} \tag{3.27}$$

$$\mu_{sl}^{(0)} = \frac{exp\left(f_{sl}^{node}\right)}{\sum_{l'} exp\left(f_{sl'}^{node}\right)} \tag{3.28}$$

$$\mu_{sl}^{(i+1)} = \frac{exp\left(f_{sl}^{node} + f_{sl}^{edge}\left(\mu^{(i)}\right)\right)}{\sum_{l'} exp\left(f_{sl'}^{node} + f_{sl'}^{edge}\left(\mu^{(i)}\right)\right)} \tag{3.29}$$

Note that the mean field inference procedure is also used at train time to approximate quantities of interest within the partial derivatives of the conditional log likelihood $L(\Gamma, \Psi)$.

3.5 Discussion

The scalar predictors of choice will be used in the following chapters as part of a no prior classifier only strategy to solve the problem of semantic segmentation applied to terrain and road terrain. The majority of works on semantic segmentation of natural environments reviewed in chapter 2 refrain from making assumptions about classes, including terrains. This is not surprising given the unstructured nature of such environments as pointed by Manduchi *et al.* (2005). Finally, within a structured output learning framework, the mere intuition that neighbouring segments of similar saliency are likely part of the same class is formalized using a probabilistic graphical a model.

Chapter 4 Scalar and structured prediction of 2D atomic units

4.1 Introduction

From an image recognition perspective the forward driving scene of a vehicle is a blend of *things* and *stuff* (Heitz and Koller 2008). *Things* are objects that have a distinctive spatial extent or shape such as cars, pedestrians, cyclists and so on. *Stuff* on the other hand embodies materials of homogeneous or granularly repetitive patterns and of amorphous spatial extent such as grass, trees, sky and so on. When a clear distinction is made between the two it becomes obvious that one needs to seek a shape pattern (Dalal and Triggs 2005) to recognise *things* and texture or colour to recognise *stuff* (Fernandez-Maloigne and Bonnet 1995), (Khan, Komma and Zell 2011). This chapter explores ways in which the texture saliency of *stuff* (particularly terrain classes) can be extracted from images of a monocular colour camera in order to achieve accurate semantic segmentation and to serve as a prerequisite for a terrain response ADAS. To this end, images are pre-segmented into superpixels (Achanta *et al.* 2012) and each of these atomic units are represented by colour, rough position and most importantly a texture pattern obtained with bag-of-visual-words. Only the latter is subject to change as local binary patterns (Ojala, Pietikainen and Maenpaa 2002), textons (Kae *et al.* 2013), (Malik *et al.* 1999) and daisy (Tola, Lepetit and Fua 2010) are among the texture descriptors computed at every pixel of an image. These flavours of textures are explored with several scalar predictors

such as support vector machine, random forest and logistic regression as well as with a structured predictor namely the conditional random field in a bid to find a winning texture and classification scheme for terrain recognition with monocular vision. The 2D analysis is performed in order to ensure that monocular vision is used effectively at an early stage. Useful guidelines can be established such as how to extract discriminative texture saliency. After all, visual saliency is the main cue needed for terrain recognition. With the prospective introduction of depth, either from stereo vision or some other ranging sensor, terrain classification is going to be more computationally demanding as more time needs to be spent extracting the relevant features and evaluating prediction hypotheses.

4.2 Related work

Similar to the work of this chapter, Khan (2013) seeks to capture texture saliency from 2D atomic units with either genuine texture descriptors or interest point descriptors in order to achieve semantic segmentation of terrain classes such as gravel, asphalt/tarmac, grass and tiles using a monocular camera. To obtain 2D atomic units for classification, Khan (2013) rigidly partitions images into grid cells at a resolution of choice. While different texture flavours are indeed explored by Khan (2013), those based on interest point descriptors such as daisy (Tola, Lepetit and Fua 2010) are sparsely computed at chosen locations (e.g. at the centre pixel of each cell) to become features. In contrast this chapter takes a similar feature extraction approach to Kae *et al.* (2013) by partitioning the image into superpixels and describing each superpixel with bag-of-visual-words (Gheorghe *et al.* 2015). Moreover the work of Khan (2013) does not tap into structured predictors with their texture flavours whereas the experiments of this chapter suggest that there is much to gain in terms of accuracy and the bulk of computational burden does not come from the structured predictor itself.

4.3 Experiments

Experiments throughout this thesis have been carried on a workstation equipped with an Intel Xeon CPU (E5-2640) having a processor base frequency of 2.50 GHz and 24 GB of RAM. Having a dataset to validate the proposed classification methods is of paramount importance. To this end, a team composed of JLR engineers has been assembled to collect images in a forward driving perspective and log them during vehicle traversal of various terrain classes. Any subsequent calculations were made using the provided data. More specifically, the proprietary JLR dataset is composed of colour images recorded in a predominantly natural environment (i.e. containing likely types of terrains experienced by on-road and off-road capable vehicles) at the Jaguar Land Rover test track facilities of Gaydon in Warwickshire, England. Terrain classes have good intra-class variability and inter-class variability. The former refers to how different the training samples corresponding to a certain class are while the latter refers to how different the classes are among themselves. Throughout data logging the weather conditions ranged from sunny to cloudy and rainy across the span of an entire afternoon. Images are pre-segmented into superpixels and the classification schemes outlined in Chapter 3 are used to predict the semantics of all the atomic units. The following subsections will clarify the various aspects of these experiments. For diagrams showing an overview of the 2D (i.e. image plane only) processing steps that a typical image undergoes in pursuit of semantics, the reader is referred to Appendix A, diagrams A.1 for scalar prediction and A.2 for structured prediction. A number of standard software libraries have been used in order to speed up experiments (as opposed to reimplementing functionality from scratch). Most notably OpenCV library (http://opencv.org) is written in C++ and provides facilities for manipulating images, extracting standard features and even machine learning (e.g. clustering and so on). Moreover, Scikit-Learn (http://scikit-learn.org/stable/) is a collection of machine learning tools written in Python language (Pedregosa *et al.* 2011). In particular, the Scikit-Learn implementation of the

SVM and RF classifier algoritms described in Chapter 3 has been utilised for experiments. The reader is reffered to Appendix B, code snippet B.2 and code snippet B.3 for basic usage example of these algorithms. The other classifier algorithms more intimately linked to probabilistic graphical models namely, LR and CRF (also described in Chapter 3) have been used as part of the GLOC Matlab code (http://vis-www.cs.umass.edu/GLOC/) developed by Kae *et al.* (2013). Again, the high level codes for the two algorithms as used throughout this thesis (for label predictions) are reproduced in Appendix B, code snippet B.4 and code snippet B.5. However, the interested reader seeking to make use of the respective LR and CRF implementations is strongly encouraged to download GLOC using the link provided and follow/make use of the various subroutines spanning training and inference aspects.

4.3.1 Superpixel segmentation

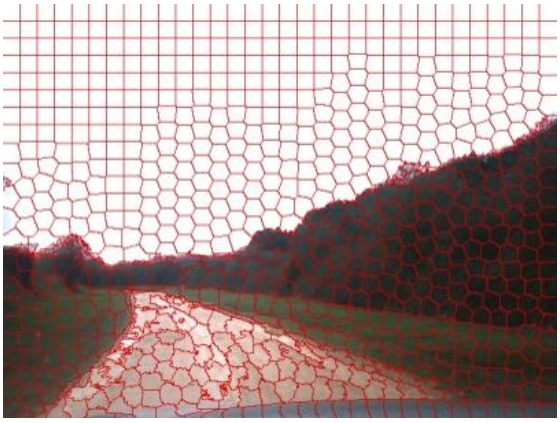


Figure 4.1 SLIC superpixels overimposed on the original image

Superpixels are regions of an image that capture redundancies by grouping locally similar pixels (Figure 4.1). They have become very popular for multi-class image segmentation (Fulkerson, Vedaldi and Soatto 2009), (Li and Sahbi 2011) and are typically used to extract features locally. They are akin to building blocks that can be assembled together to form semantic objects. Simple linear iterative clustering (SLIC) proposed by Achanta *et al.* (2012) is an algorithm capable to generate superpixels efficiently by leveraging the K-means clustering algorithm in

the Lab colour space. It begins by initialising cluster centres as image pixels on a regular grid. These seed points are S pixels apart.

$$C_k = [L_k, a_k, b_k, x_k, y_k]^T \tag{4.1}$$

The K-means search is restricted to a region of at most 2S x 2S centred around C_k. Pixels are assigned to clusters based on a distance measure designed to reflect both spatial and colour proximity as well as the relative importance between the two. In addition to being fast and memory efficient, SLIC has been found to exhibit good adherence to image boundaries compared to other superpixel methods (Achanta *et al.* 2012). SLIC superpixels will be the 2D atomic units through this entire work.

4.3.2 Features

Features are the lenses through which a learning algorithm will see the world, and therefore such features should be carefully selected in order to obtain good discrimination between classes of interest, regardless of the problem domain. This has traditionally been the case due to limited amounts of training data and processing burden. However, such paradigm is gradually being waived as more data becomes available and computation capabilities develop. In this work, feature representation will be considered in the classical hand engineered sense. Node features comprised of colour, position and texture are a suitable representation for every superpixel and discriminative learning is able to leverage such representation. In addition to node features, edge features between adjacent superpixels capture context and can improve the accuracy when cast within a structured output learning framework (Figure 4.2). Such edge features are typically designed to preserve discontinuities in labelling of the atomic units. They include boundary information as well as measures of colour and texture dissimilarity.

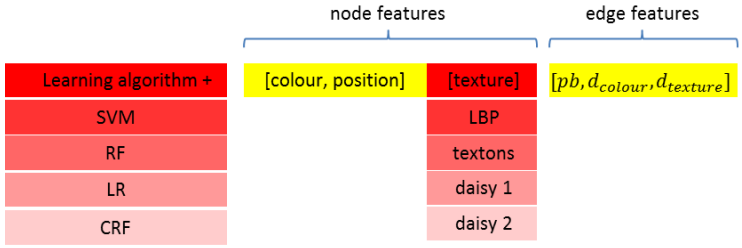


Figure 4.2 Overview of the configurable models. Algorithms and features (seen in red boxes) are explored while the other components (seen in yellow boxes) remain fixed. Texture features can be toggled to use texture descriptors such as local binary patterns (LBP), filter responses (textons) or two quantization levels of daisy key points. Edge features such as probability of boundary (pb), colour distance (d_{colour}) and texture distance ($d_{texture}$) only apply to structured prediction using the CRF.

4.3.2.1 Node features

- **Colour histogram**

  Colour histogram takes the bag-of-words approach to describe each superpixel as a bin count of colour prototypes in the Lab colour space. First a number of images are selected from the training set and then clustering is applied in this space on the image pixels using K-means with 64 seed points. Unlike the standard algorithm, cluster initialisation is done in a probabilistic manner as suggested by Arthur and Vassilvitskii (2007). After convergence, each pixel can be assigned one of K prototypes depending on its proximity. The dimension of the colour histogram is tied to 64 bins for all feature representations of superpixels.

- **Position**

  Partitioning the image into an 8 x 8 grid allows for each superpixel to have a degree of membership to its cells (Figure 4.3). A normalised histogram over the 64 bins is computed to incorporate preliminary spatial cues from the image domain.

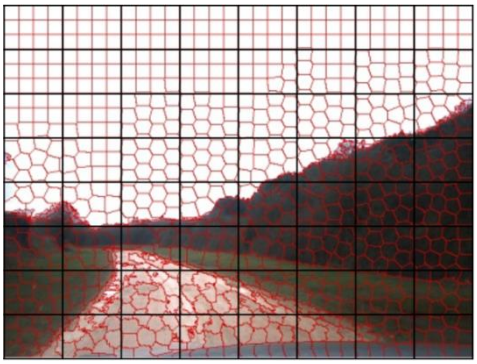


Figure 4.3 Position bins. Image is divided into 64 cells with the same aspect ratio. Superpixel position features represent the normalised superpixel distribution across the cells.

- **Texture histograms**

  In order to capture the texture information of every superpixel a number of popular texture descriptors will be used in a bag-of-words fashion. The impact on the classification accuracy in 2D will be the main decision factor in selecting a suitable descriptor towards further model improvements with 3D information.

  Local binary patterns

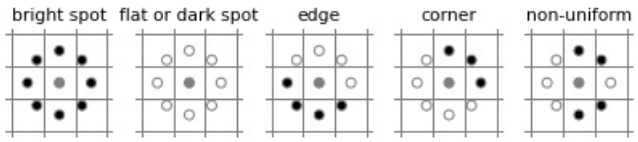


Figure 4.4 Typical unit radius, 8 points LBP with {0, 1, origin} as {black, white, grey} circles

Local binary patterns (Ojala, Pietikainen and Maenpaa 2002) are simple and efficient texture descriptors (Figure 4.4) that are robust to illumination changes and can be made rotationally invariant. Initially introduced for grayscale images, recent years have seen many variants of LBP emerge including descriptors that consider the colour channels (Zhu, Bichot and Chen 2010). They have successfully been used for a wide range of applications such as human detection (Wang, Han and Yan 2009), face and expression

recognition (Ahonen, Hadid and Pietikainen 2006), (Huang *et al.* 2011), (Liao et al. 2006), pavement crack detection (Hu and Zhao 2010), visual terrain classification for robotic applications (Khan 2013), (Khan, Komma and Zell 2011) and more generally texture analysis (Ojala, Pietikainen and Maenpaa 2000), (Zolynski, Braun and Berns 2008). At the most basic level the operator creates a binary pattern by comparing a central pixel or origin with its P neighbours sampled on a circle of radius R. Whenever the sampled points do not fall in the centre of a pixel the comparison is done with bi-linearly interpolated pixel values. A total of 2^P binary sequences are possible.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) * 2^p, \qquad s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{4.2}$$

To compensate for image rotations that will inevitably change the operator's output, binary sequences are shifted like a "rotary dial" such that a maximum number of bits are 0 beginning with the most significant. Out of these, the uniform patterns are fundamental properties of texture accounting for around 90% of all patterns (Ojala, Pietikainen and Maenpaa 2002). For a pattern to be considered uniform it should contain no more than two transitions between 1 and 0 (Figure 4.5, line 1).

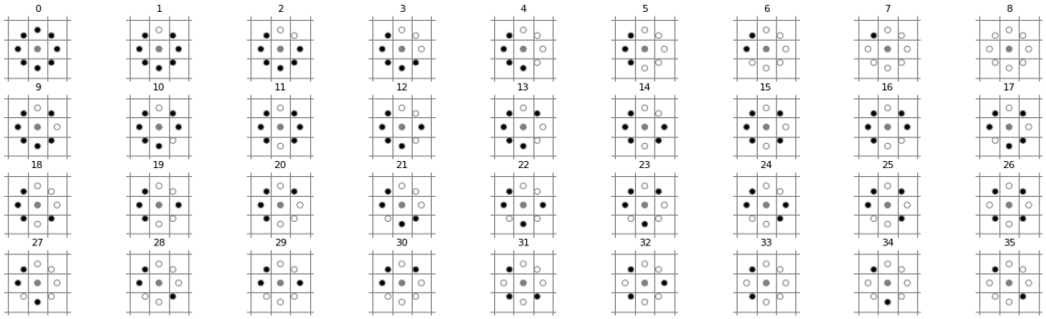


Figure 4.5 For P=8 there are 36 unique rotation invariant patterns; first 9 patterns are uniform and the rest are non-uniform

$$U(LBP_{P,R}) = |s(g_{P-1} - g_c)| + \sum_{p=1}^{P-1} |s(g_p - g_c) - s(g_{p-1} - g_c)| \tag{4.3}$$

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) \ if \ U(LBP_{P,R}) \leq 2 \\ P + 1 \ otherwise \end{cases} \quad (4.4)$$

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (4.5)$$

Vector quantization is done implicitly by simply assigning the pixels with a coded value ranging from 0 to 8 for the uniform patterns and 9 for all the others. This coding paves the way for describing the texture of each superpixel as a normalised histogram of prototypes spread across 10 bins.

Dense daisy pixel descriptors

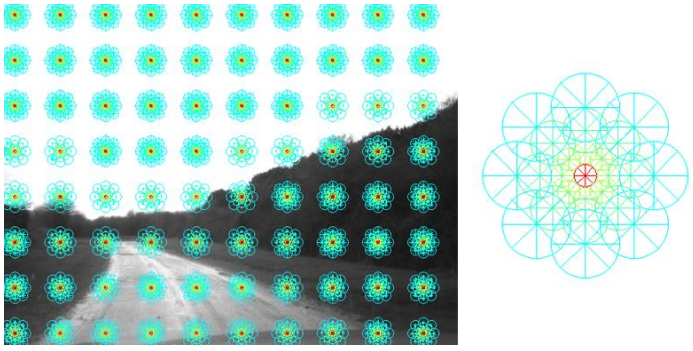


Figure 4.6 Sparse daisy pixel descriptors for visualisation purpose. Circle radius is proportional to the amount of Gaussian smoothing at different directions when computing the histogram

Daisy pixel descriptors (Figure 4.6) have been introduced by Tola, Lepetit and Fua (2010) in the context of stereo vision research to enable the computation of dense depth and occlusion maps from image pairs with a wide baseline. Since then, they have been applied to solve problems outside their initial scope such as visual object recognition (Chao, Bichot and Liming 2011) and face recognition (Velardo and Dugelay 2010). Similar to SIFT (scale invariant feature transform) (Lowe 2004), daisy relies on

gradient orientation histograms too. However, for descriptor use in dense scenarios daisy can be computed much faster than SIFT. Orientation maps $M_o = (\frac{\partial I}{\partial o})^+$ can be computed given an image I for every quantized direction o by keeping only the positive values of the image gradient norm in order to preserve the polarity of the intensity changes. Each of these orientation maps is convolved with Gaussian kernels of increasing $\sum$ to form convolved orientation maps.

$$G_o^{\sum} = N_{\sum} * M_o \tag{4.6}$$

A great computational speedup is achieved here due to the fact that larger kernel convolutions can be obtained from smaller consecutive ones. Daisy descriptors use a circular grid as location for histogram calculations from all the values of $G_o^{\sum}$ at that certain orientation. This is in contrast to the regular grid used in SIFT as well as to the fact that SIFT relies on a triangular shaped kernel. Concentric circles around the pixel location from the convolved orientation maps contribute with values to make up the descriptor. The final descriptor is a concatenation of normalised histograms. In its default setting the daisy descriptor will assign each pixel with a 200D vector resulting in 640 × 480 descriptors for each image present in the JLR data set. Again the bag of words approach is taken in order to have a fixed size feature vector to describe each superpixel. Firstly a number of training images have been set aside for clustering using an improved K-means via probabilistic seeding (Arthur and Vassilvitskii 2007). Due to the fact that daisy descriptors can be computed densely and efficiently while being competitive with SIFT in terms of performance, two sets of prototypes have been generated and tested: daisy 1 and daisy 2. For daisy 1, K-means has been run with 64 clusters while for daisy 2 with 300 clusters or prototypes. As for vector quantization, each superpixel texture has been represented as a normalised histogram of bin counts based

on prototype proximity. Assigning each descriptor to its closest prototype was based on the Euclidean distance.

Textons

The literature describing textons is somewhat ambiguous (Zhu *et al.* 2005) but loosely the concept of textons refers to fundamental micro-structure or texture building blocks in images. Malik *et al.* (1999) define textons as frequently co-occurring combinations of oriented linear filter outputs that can be learned using a K-means approach. It is this procedure that is appealing and enables each superpixel to be described by a normalised histogram of prototypes. Similar to Kae *et al.* (2013), Malik *et al.* (1999), a number of training images specific to each class are convolved with a bank of filters to get a vector of 36 responses for each pixel. These responses correspond to 12 filters of different orientation taken at 3 different scales. K-means with probabilistic cluster initialisation (Arthur and Vassilvitskii 2007) has been used generate 64 prototype vectors of filter responses as cluster centroids. Again, each superpixel is represented by a normalised histogram of texton counts based on the proximity of filter responses to their prototype vectors.

4.3.2.2 Edge features

Edge features have been used only for structured classification using the conditional random field (CRF):

- Probability of Boundary

  In $P(b)$ images (Martin, Fowlkes and Malik 2002), each pixel of the original image is assigned a probability of belonging to a boundary (Figure 4.7). Probabilities that lie on the border of adjacent SLIC superpixels are summed to form an edge feature. Intuitively, superpixels with an image boundary between them are more likely to have different labels.

Figure 4.7 Probability of boundary image. White indicates high probability to have a boundary and conversely darker regions are assigned lower probabilities

- Colour

  After segmenting the images with SLIC and converting them to Lab colour space, the mean colour can be computed for each superpixel.

$$\overline{Lab} = \frac{1}{n}\sum_{i \in sp} L_i, a_i, b_i \tag{4.7}$$

  The l_2 distance is taken between the mean Lab colours of neighbouring superpixels. The more dissimilar their average colours, the more likely the superpixels are to have different labels.

$$d_{colour} = \sqrt{\sum_{i=1}^{3} (\overline{Lab}_1(i) - \overline{Lab}_2(i))^2} \tag{4.8}$$

- Texture

  The chi-squared distance between texture histograms h_1 and h_2 of neighbouring superpixels is computed similar to Huang, Narayana and Learned-Miller (2008), Kae *et al.* (2013). Texture histograms may have different dimensions depending on the texture feature and the number of prototypes p used to describe each superpixel. Again, the

more dissimilar their texture, the more likely the superpixels are to have different labels.

$$d_{texture}(h_1, h_2) = \frac{1}{2}\sum_{i=1}^{p}\frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)}, \qquad p = \begin{cases} 64\ textons \\ 10\ LBP \\ 64\ daisy\ 1 \\ 300\ daisy\ 2 \end{cases} \tag{4.9}$$

4.3.3 Evaluation

In order to quantify the performance of structured prediction versus independent as well as the appropriateness of appearance measures (e.g. colour, position, texture and boundaries), a number of popular measures (Sokolova and Lapalme 2009) from the field of semantic labelling have been used in conjunction with the JLR dataset.

4.3.3.1 JLR dataset and classification

- Images

Table 4.1 Typical training and testing images present in the JLR dataset

A total of 430 images at a resolution of 640 × 480 where used during the experiments (Table 4.1). They have been recorded to contain terrain scenes and road scenes, with as little other human made structures as possible (i.e. predominantly natural environment). Different illumination conditions as well as good intra-class and inter-class variability are considered. Out of the total number of images 50% where used for training, 20% for validation and 30% for testing. After initialising the SLIC parameters such as

the region size and a suitable trade-off between appearance and spatial regularity, each image has been partitioned into approximately 768 superpixels. While this is still an unbalanced dataset (due to omnipresent classes such as sky or grass), having a fine superpixel representation of each image is twofold. Firstly this ensures that real world boundaries are not violated by coarser superpixels and secondly it allows for a fairly substantial amount of samples representative for each class.

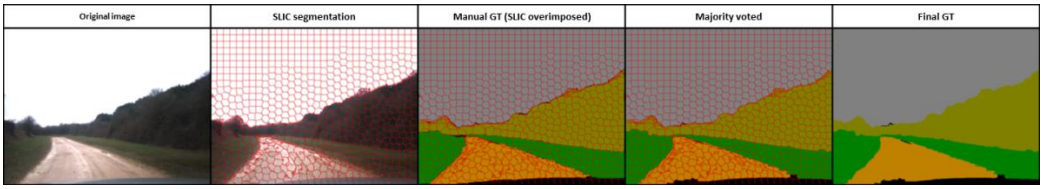
- Ground truth

Table 4.2 Colour coded labels using RGB values

R, G, B	label
0, 128, 0	grass
128, 128, 0	tree
128, 128, 128	sky
192, 128, 0	dirt
128, 64, 0	gravel
0, 192, 0	shrubs
128, 64, 128	tarmac
0, 0, 0	void

Pixels of original images have been manually labelled and colour coded (Table 4.2) using Gimp image editor (https://www.gimp.org/). In order to obtain superpixel rather than pixel ground truth, manually annotated images have been overimposed on SLIC segmented images. Each superpixel received a ground truth depending what label it spanned most via a majority vote strategy (Table 4.3). Subsequent training and evaluation of the proposed methods took place entirely in the superpixel domain.

Table 4.3 From manually annotated pixel-wise ground truth (GT) to the superpixel ground truth used for evaluation

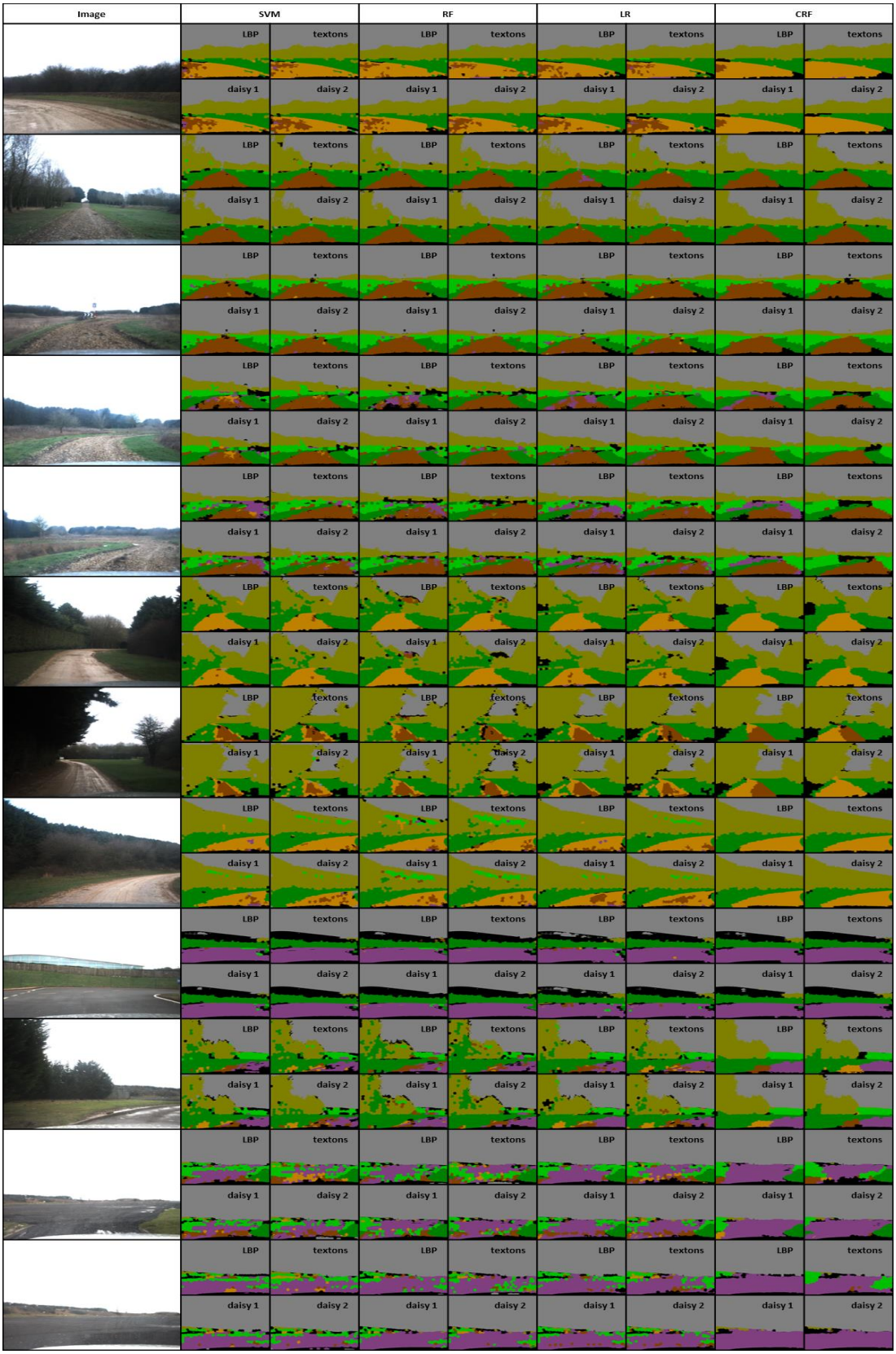


- Classification results

In order to show visually how each feature behaves under a different discriminative learning algorithm colour coded results are presented (Table 4.4). Structured prediction with a CRF is able to filter out the outliers (or inconsistent patches of a certain class) and produce a smooth output without violating boundary constraints. Contextual information is therefore important as isolated patches are possible but unlikely given the CRF parameterisation. On the other hand, the choice of a suitable texture descriptor is also important. For example, structured prediction based on descriptors that do not capture texture intricacies yields lower performance than independent (i.e. scalar) superpixel predictions based on descriptors that do. This becomes more obvious by inspecting Table 4.6 and Table 4.7. The scores present in these tables have been manually inputted after evaluating the classifiers described in Chapter 3 against their respective feature variants. The entries in both tables are sorted according to the overall superpixel classification accuracy obtained on the test data. Prior to these tests, the algorithms have been trained using the training subset of the JLR dataset and depending on the case some parameters have been further tuned using the validation subset. To compute the individual per-class accuracy, entries corresponding to recall (Table 4.6) and intersection-over union (Table 4.7), one needs to determine the amounts of true positives, false negatives and false positives. Such quantities are more easily understood and obtained from a confusion matrix. The following subsections will clarify these aspects. Furthermore, it will become more apparent why intersection-

over-union is a better suited accuracy measure given its ability to penalise both over-estimation and under-estimation. As basic texture descriptors, LBP have been found to perform poorly on the JLR dataset in comparison to the other descriptors. One possible explanation for these phenomena would be the quantisation mechanism assigning the same value for all non-uniform patterns and therefore restricting texture dissimilarity at an early stage. Furthermore, the standard LBP with $P = 8$ neighbours around a central pixel resulted in a superpixel representation as a histogram of just 10 prototypes. This is a fairly coarse texture resolution. In comparison, texture components across the other superpixel features are modelled by occurrences of at least 64 prototypes. Having inspected Table 4.7 it is easy to envision how a certain model (i.e. algorithm and visual saliency feature) would better serve in a particular scenario. For example, terrain classes such as {dirt, gravel, shrubs and tarmac} are more accurately classified using CRF + daisy2. Along with the overall superpixel accuracy (i.e. from the least accurate to the most), the processing times of various stages ranging from feature extraction all the way to prediction, are detailed in Table 4.8. Such processing times are best visualised in relative terms as in Figure 4.11.

Table 4.4 Successful segmentation results on the JLR Dataset. Textons (quadrant 1), LBP (quadrant 2), daisy 1 (quadrant 3) and daisy 2 (quadrant 4) under different discriminative learning algorithms: SVM, RF, LR and CRF



4.3.3.2 Overall superpixel accuracy

The overall superpixel accuracy is the fraction of all superpixels in the test set that have been correctly identified.

$$overall\ acc\left(y_{true}, y_{predicted}\right) = \frac{1}{n_{superpixels}} \sum_{i=1}^{n_{superpixels}} 1\left(y_{predicted} = y_{true}\right) \quad (4.10)$$

$$1(condition) = \begin{cases} 1\ if\ condition = true \\ 0\ if\ condition = false \end{cases} \quad (4.11)$$

4.3.3.3 Confusion matrix

A confusion matrix allows visualisation of the model performance in the form of a table depicting actual labels and predictions. For a given true class within the test set the confusion matrix shows how likely any of the possible labelling is. It is therefore a strong indicator that the feature space as parameterised (or partitioned) by a learning algorithm might be overlapping between certain classes (Figure 4.8). For sample code on how to generate a confusion matrix using Scikit-Learn in Python language (Pedregosa *et al.* 2011) such as those present in Figure 4.8 see Appendix B, code snippet B.1.

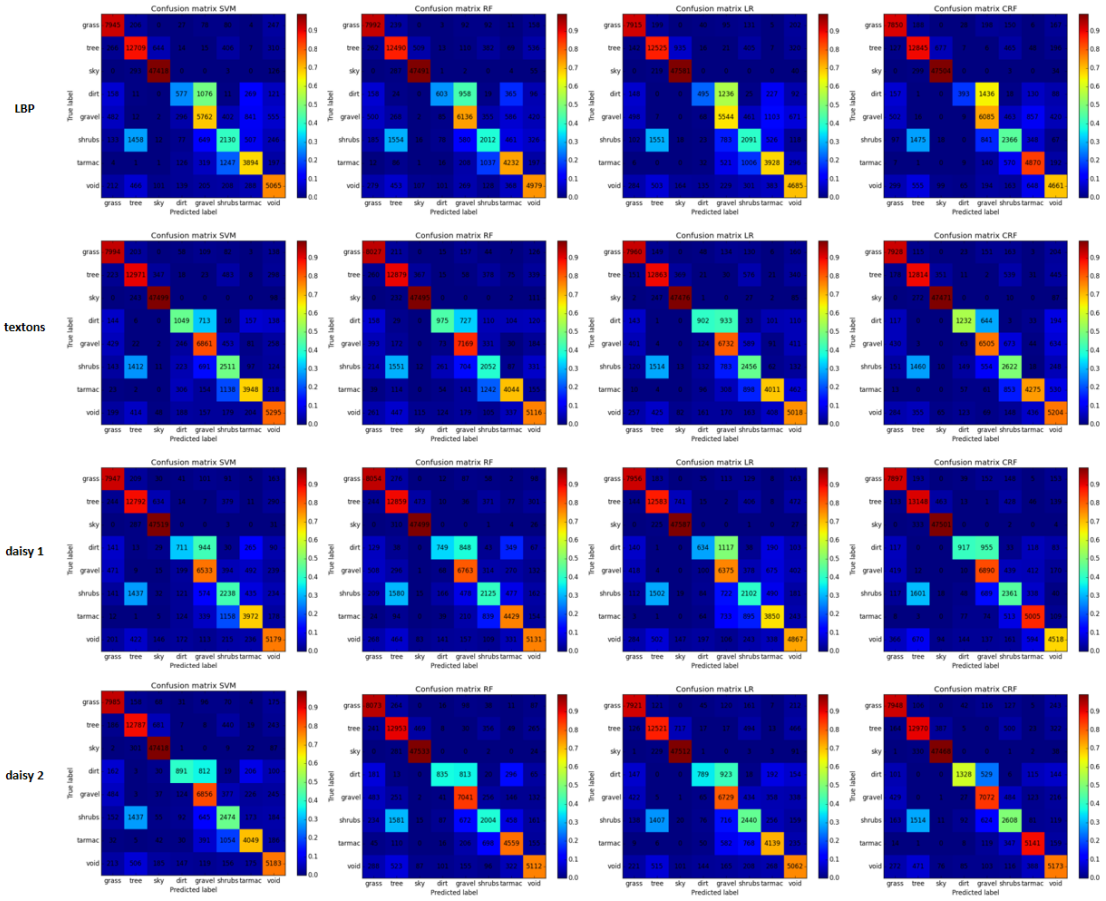


Figure 4.8 Confusion matrices obtained with different configurations of discriminative algorithms and texture representations. Structured prediction has the advantage of smoothing out some previously misclassified examples given the superpixel dissimilarity measures. In particular, the chi-squared difference x^2 is able to produce a significant improvement with daisy 2.

4.3.3.4 *Error reduction*

Error reduction is computed with respect to a baseline, in this case the accuracy of SVM in conjunction with textons obtained as bag-of-visual-words from filter responses in the image domain.

$$E(model) = \frac{[100 - acc(baseline)] - [100 - acc(model)]}{100 - acc(baseline)} \times 100 \qquad (4.12)$$

4.3.3.5 *Recall, Intersection-over-union*

Confusion matrix entries are classified into four categories: true positives (TP), false negatives (FN), true negatives (TN) and false positives (FP). These entries can be used to evaluate the accuracy (recall and intersection-over-union) of each class. Given a particular class j, true positives (TP_j) are those manually labelled examples of j that have been correctly identified. The false negatives (FN_j) are those manually labelled examples of j that should have been correctly identified. True negatives (TN_j) are all the other manually labelled examples that have been correctly classified. False positives (FP_j) are those predicted j examples that disagree with the ground truth. For example if grass is under scrutiny, the following table of confusion (Table 4.5) can be created.

Table 4.5 Confusion for {grass}

true label	**grass**	$\boldsymbol{TP}_{grass}$ (actual grass that was correctly classified as grass)	$\boldsymbol{FN}_{grass}$ (grass that was incorrectly classified as other e.g. tree, sky, dirt etc.)
	other	$\boldsymbol{FP}_{grass}$ (other classes e.g. tree, sky, dirt etc. that were incorrectly classified as grass)	$\boldsymbol{TN}_{grass}$ (all the other classes correctly classified as non-grass)
		grass	**other**
		predicted label	

For each class recall accuracy is defined as the number of superpixels correctly classified by the learning algorithm divided by the number of superpixels being tested. In other words recall is the correctly labelled fraction of the ground truth.

Another per-class accuracy measure (i.e. intersection-over-union) is obtained as the number of superpixels correctly identified divided by the union between the ground truth and predicted superpixels. Subsequently, the average of each measure is computed as the arithmetic mean of individual class accuracies (i.e. the macro-average).

$$recall_j = \frac{TP_j}{TP_j + FN_j} \tag{4.13}$$

$$\overline{recall} = \frac{1}{m_{classes}} \sum_{j=1}^{m_{classes}} recall_j \tag{4.14}$$

$$intersection/union_j = \frac{TP_j}{TP_j + FN_j + FP_j} \tag{4.15}$$

$$\overline{intersection/union} = \frac{1}{m_{classes}} \sum_{j=1}^{m_{classes}} intersection/union_j \tag{4.16}$$

Intersection-over-union is however a more appropriate accuracy measure than recall under the given scenario. In the following example (Figure 4.9) if the class “void” is under scrutiny the recall measure yields maximum accuracy thereby favouring over-estimation. Conversely, the intersection-over-union assigns a low accuracy to “void” thus allowing for an independent per class measurement by penalising both over-estimation and under-estimation (Figure 4.10).

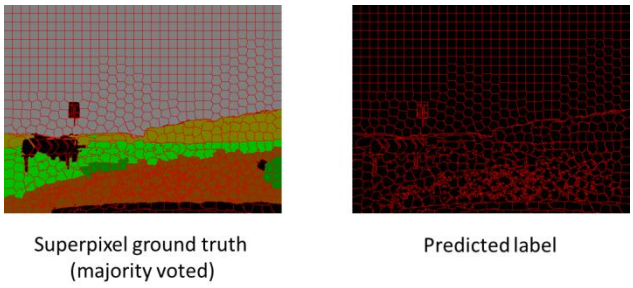


Figure 4.9 Hypothetical labelling

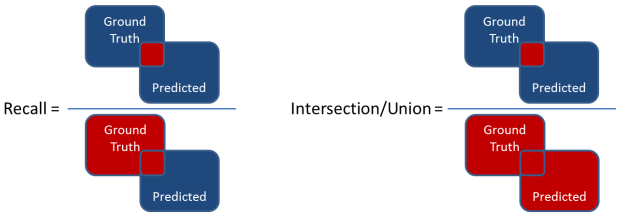


Figure 4.10 Recall and intersection-over-union accuracy measures

Table 4.6 Recall accuracies; models are sorted in ascending order of their overall accuracy

model	overall acc	E(model)	grass	tree	sky	dirt	gravel	shrubs	tarmac	void	average recall
LR + [LBP]	85.57	-30.83	92.17	87.15	99.46	22.27	66.38	40.12	67.85	70.09	68.19
SVM + [LBP]	86.31	-24.12	92.52	88.44	99.12	25.96	68.99	40.87	67.27	75.78	69.87
RF + [LBP]	86.75	-20.13	93.07	86.91	99.27	27.13	73.47	38.60	73.10	74.49	70.76
LR + [daisy 1]	86.77	-19.95	92.65	87.56	**99.47**	28.52	76.33	40.33	64.86	74.45	70.52
CRF + [LBP]	87.40	-14.23	91.42	89.38	99.30	17.68	72.86	45.40	84.13	69.73	71.24
SVM + [daisy 1]	87.72	-11.33	92.55	89.01	99.33	31.98	78.22	42.94	68.61	77.48	72.52
LR + [daisy 2]	87.94	-9.34	92.24	87.13	99.31	35.49	80.57	46.82	71.50	75.73	73.60
LR + [textons]	88.25	-6.53	92.70	89.51	99.24	40.58	80.60	47.12	69.29	75.07	74.26
RF + [daisy 1]	88.44	-4.81	93.79	89.48	99.29	33.69	80.97	40.77	76.51	76.77	73.91
SVM + [daisy 2]	88.48	-4.44	92.99	88.98	99.12	40.08	82.09	47.47	69.94	77.54	74.78
RF + [textons]	88.59	-3.45	93.48	89.62	99.28	43.86	**85.84**	39.37	69.86	76.54	74.73
CRF + [textons]	88.89	-0.73	92.33	89.17	99.23	55.42	77.89	**50.31**	73.85	77.86	77.00
RF + [daisy 2]	88.95	-0.18	**94.01**	90.13	99.36	37.56	84.30	38.45	78.75	76.48	74.88
SVM + [textons]	88.97	0	93.09	90.26	99.29	47.19	82.15	48.18	68.20	**79.22**	75.95
CRF + [daisy 1]	89.08	1.00	91.96	**91.49**	99.29	41.25	82.50	45.30	86.46	67.59	75.73
CRF + [daisy 2]	**90.56**	**14.42**	92.56	90.25	99.22	**59.74**	84.67	50.04	**88.81**	77.39	**80.34**

Table 4.7 Intersection-over-union accuracies; models are sorted in ascending order of their overall accuracy

model	overall acc	E(model)	grass	tree	sky	dirt	gravel	shrubs	tarmac	void	average ∩/∪
LR + [LBP]	85.57	-30.83	81.04	74.33	97.19	19.51	49.34	27.73	48.86	55.62	56.70
SVM + [LBP]	86.31	-24.12	80.73	75.57	97.57	19.88	53.88	28.14	50.54	59.69	58.25
RF + [LBP]	86.75	-20.13	80.06	72.27	97.97	23.93	58.05	27.85	55.30	58.77	59.27
LR + [daisy 1]	86.77	-19.95	82.12	74.95	97.62	23.33	57.20	28.79	50.36	59.88	59.28
CRF + [LBP]	87.40	-14.23	80.30	75.98	97.68	16.79	54.49	33.59	62.23	59.39	60.06
SVM + [daisy 1]	87.72	-11.33	81.12	76.37	97.51	24.57	62.64	29.91	54.91	65.48	61.56
LR + [daisy 2]	87.94	-9.34	82.07	75.18	97.60	30.10	61.88	33.43	60.11	60.70	62.64
LR + [textons]	88.25	-6.53	**82.31**	76.95	98.29	32.15	62.86	32.20	61.90	59.85	63.31
RF + [daisy 1]	88.44	-4.81	80.79	73.78	98.11	28.17	66.51	30.59	60.68	67.30	63.24
SVM + [daisy 2]	88.48	-4.44	81.33	76.19	96.89	33.56	65.78	33.72	61.22	65.57	64.28
RF + [textons]	88.59	-3.45	80.98	75.20	98.26	35.26	69.48	27.65	62.88	63.55	64.16
CRF + [textons]	88.89	-0.73	81.23	77.30	98.35	46.51	66.15	34.50	67.28	57.66	66.12
RF + [daisy 2]	88.95	-0.18	80.26	74.47	98.18	33.51	68.19	30.01	64.47	**67.50**	64.57
SVM + [textons]	88.97	0	82.01	**77.80**	**98.45**	32.16	67.27	33.20	62.28	66.55	64.96
CRF + [daisy 1]	89.08	1.00	81.02	76.52	98.11	35.90	66.51	34.04	68.54	61.20	65.23
CRF + [daisy 2]	**90.56**	**14.42**	81.68	77.23	98.25	**53.51**	**71.85**	**38.39**	**78.78**	65.27	**70.62**

4.3.3.6 *Processing time*

Table 4.8 Prediction times of various models on a JLR image frame

model	feature dimension		superpixels per frame (on average)	features extraction per frame (on average)						features evaluation per frame (Matlab/Python)	total time per frame (640×480)
				all nodes			all edges				
	node	edge		colour (C++)	position (C++)	texture (Matlab)	pb (Matlab)	d_{colour} (C++)	$d_{texture}$ (C++)		
LR + [LBP]	138	n/a	768	0.164	0.001	0.184	n/a	n/a	n/a	0.001	0.350
SVM + [LBP]	138	n/a	768	0.164	0.001	0.184	n/a	n/a	n/a	4.688	5.037
RF + [LBP]	138	n/a	768	0.164	0.001	0.184	n/a	n/a	n/a	0.044	0.393
LR + [daisy 1]	192	n/a	768	0.164	0.001	2.167	n/a	n/a	n/a	0.001	2.333
CRF + [LBP]	138	3	768	0.164	0.001	0.184	60.658	0.002	0.005	0.949	61.963
SVM + [daisy 1]	192	n/a	768	0.164	0.001	2.167	n/a	n/a	n/a	6.212	8.544
LR + [daisy 2]	428	n/a	768	0.164	0.001	2.806	n/a	n/a	n/a	0.001	2.972
LR + [textons]	192	n/a	768	0.164	0.001	1.364	n/a	n/a	n/a	0.001	1.530
RF + [daisy 1]	192	n/a	768	0.164	0.001	2.167	n/a	n/a	n/a	0.045	2.377
SVM + [daisy 2]	428	n/a	768	0.164	0.001	2.806	n/a	n/a	n/a	13.627	16.598
RF + [textons]	192	n/a	768	0.164	0.001	1.364	n/a	n/a	n/a	0.045	1.574
CRF + [textons]	192	3	768	0.164	0.001	1.364	60.658	0.002	0.007	0.905	63.101
RF + [daisy 2]	428	n/a	768	0.164	0.001	2.806	n/a	n/a	n/a	0.069	3.040
SVM + [textons]	192	n/a	768	0.164	0.001	1.364	n/a	n/a	n/a	5.781	7.310
CRF + [daisy 1]	192	3	768	0.164	0.001	2.167	60.658	0.002	0.007	0.886	63.885
CRF + [daisy 2]	428	3	768	0.164	0.001	2.806	60.658	0.002	0.011	0.888	64.530

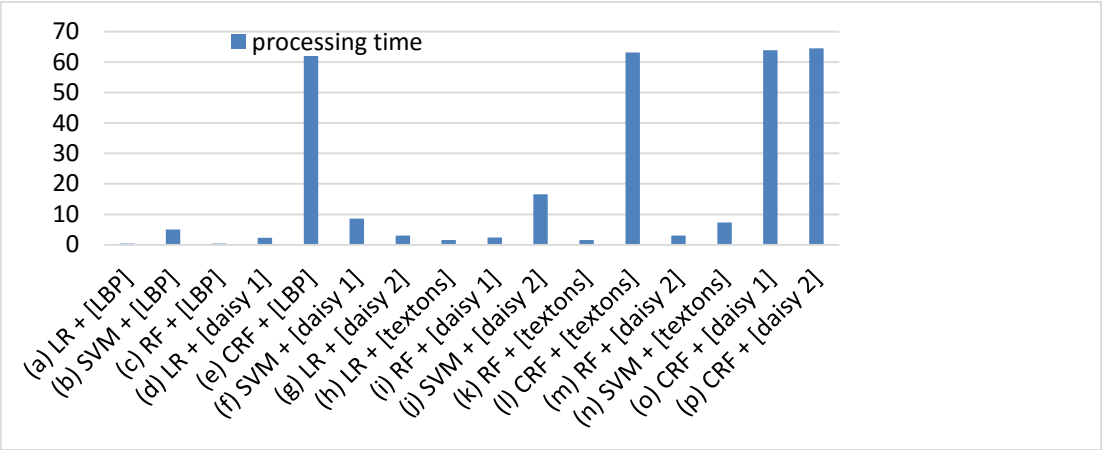


Figure 4.11 Processing times should be observed in relative terms as test platform and image resolution might change

Processing times are measured for various steps ranging from feature extraction all the way to prediction in order to identify where the bulk of computational burden comes from (Table 4.8). Pinpointing processing times in this fashion encourages future method optimisation to consider a trade-off between accuracy and computation demands. For the time being, it is obvious that the structured predictor improves the accuracy given the right texture saliency. While the total processing time for an image frame under such model is seen to soar (Figure 4.11), most of this time is spent extracting one of the discontinuity preserving features, namely the probability of boundary (Martin, Fowlkes and Malik 2002).

4.3.4 Comparison to prior work

SVM and bag-of-visual-words has been used for benchmarking (SVM + [textons]), a model that resembles the work of Filitchkin and Byl (2012), where terrain was classified into {tarmac, grass, gravel, mud, soil and woodchips}. In addition to this baseline, the model RF + [textons] resembles the work of Angelova *et al.* (2007) where colour and textons histograms are used in conjunction with an ensemble classifier. Angelova *et al.* (2007) use a variable length representation for terrain patch classification into classes such as {sand, soil, grass, gravel, tarmac, woodchips and mixed}.

4.4 Discussion

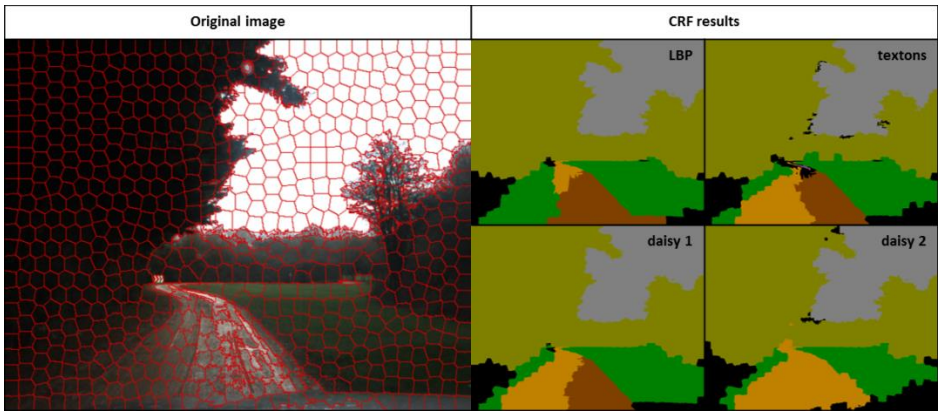


Figure 4.12 CRF results with different texture. Best results are obtained with daisy 2

In this chapter terrain recognition for driver assistance applications has been casted as a semantic segmentation task. Several texture saliency flavours have been tested with discriminative learning schemes in a bid to use monocular vision to maximum advantage. A method to segment the road scene in front of a vehicle has been fielded based on the well-established framework of discriminative graphical models. Good results have been obtained using a conditional random field (CRF) with edge potentials based on histograms of daisy texture descriptors with a large number of prototypes. Edge potentials are able to leverage on the increased texture granularity of daisy descriptors via the chi-squared measure. This provides for higher accuracy by considering a more refined dissimilarity measure between neighbouring superpixels. Gains are particularly reflected by the ability to discriminate between classes with more subtle texture differences such as fine gravel and dirt. The two classes are likely to coexist but it comes down to exceeding a certain dissimilarity threshold before accepting non smooth patches within an image region. Experimental evidence suggests that the selection of appropriate texture descriptors as well as fine representation with a larger number of prototypes (Figure 4.12) for vector quantization acts as an accuracy bottleneck. This is one of the key findings that have emerged from experimenting with different quantisation levels of daisy

as texture. Other novel aspects presented in this chapter include pairing structured output learning with bag of daisy prototypes as texture features to label superpixels of terrain classes. It builds on top of previous work presented by Gheorghe *et al.* (2015). This time both scalar and structured output learning algorithms are tested with bag of daisy prototypes. Another notable distinction of this chapter with respect of Gheorghe *et al.* (2015) is the terrain visual saliency variant obtained using bag of textons (i.e. filter responses).

.

Chapter 5 Scalar and structured prediction using 3D information

5.1 Introduction

The success of a number of ADAS applications (Geronimo *et al.* 2010), (Vahidi and Eskandarian 2003) is owed to vehicle's ability to tap into range information with the aid of a ranging sensor. This chapter explores ways to incorporate 3D surface saliency towards refined semantic segmentation of terrain types (i.e. *stuff*) using stereo vision. To this end the top performing texture saliency devised in Chapter 4 using daisy (Tola, Lepetit and Fua 2010) is kept fixed while the attention is shifted to exploring terrain surface saliency alongside both scalar predictors and the structured predictor of choice. Since stereo reconstructed point clouds spanning large distances are sparse, they are not easily partitioned into segments with standard supervoxelization methods (Douillard *et al.* 2011b), (Papon *et al.* 2013), (Xu and Corso 2012). To overcome this problem a direct correspondence is assumed between the SLIC generated superpixels (Achanta *et al.* 2012) on the left (reference) image and the real world subspaces they span when reconstructed with stereo. This leads to a pseudo-supervoxelization of the entire point cloud. The atomic units can be regarded as supervoxels altogether or as superpixels augmented with corresponding point cloud information. Two surface saliency flavours are proposed both of which are obtained with bag-of-spatial-words, summarizing the surface statistics of every atomic unit. One uses the fast point feature histograms (FPFH) of Rusu, Blodow and

Beetz (2009) and the other uses the height coordinates of point clouds. Again all classification schemes of Chapter 3 are employed in order to establish how to use stereo vision to maximum advantage. Note that the main developments of this chapter are not tied to passive ranging with stereo but generally applicable to more accurate ranging sensors, if calibrated.

5.2 Related work

Terrain recognition using a ranging sensor has been cast as a semantic segmentation task in the work of Lalonde *et al.* (2006) by considering LIDAR voxels as atomic units and classifying them into three classes, namely scatter, linear and surface. The scatter class amounts to porous volumes such as grass and foliage, linear amounts to small tree trunks and thin branches while surface amounts to ground, rocks and large tree trunks. Saliency features are extracted from covariance matrices of point neighbourhoods. Subsequently these spatial statistics are fitted by Gaussian mixture models (GMM). Similarly, in this chapter one of the saliency flavours of terrain classes is obtained using statistics of point descriptors (Rusu, Blodow and Beetz 2009) based on normals which in turn are obtained from covariance matrices of point neighbourhoods. However the atomic units differ, individual terrain classes are modelled explicitly and classification schemes are discriminative in nature. In addition the saliency features incorporate visual appearance and are tested with both scalar and structured output learning whereby smooth labelling is enforced.

5.3 Stereo vision

Stereo cameras have emerged as a cheap and often simple solution to extract 3D information in front of the vehicle. In essence, they are comprised of two (or more) slightly offset cameras by a baseline that undergoes further calibration and rectification to compensate for image rotation and distortion. After image rectification, pixel correspondence between pairs of imag-

es (Figure 5.1) is established line by line. This is due to the fact that coinciding left and right camera planes lead to simplified epipolar geometry (Grewe and Kak 1994). Similar to human vision (left and right eye), it works by triangulating points based on their 2D projections in left and right images (Figure 5.2). For every projected point, the amount of pixel shift between the two images is inversely proportional to the distance from it. Since digital images are discrete representations of continuous images, the smallest possible disparity is one pixel. In practice disparity map is computed with sub-pixel accuracy. This limiting factor affects the reconstruction of points that are far away. Close range points reconstruction is typically accurate and dense. However, as distance increases reconstruction becomes sparse and the accuracy deteriorates rapidly (Figure 5.3). Stereo vision is in itself a vast field of research involving hardware as well as software development. Finding disparity between pixels in left and right image is a hard task due to the fact that correlation schemes can find false correspondences. This is particularly problematic in non-textured images or images with repetitive and ambiguous texture.

Bumblebee2 stereo rig is factory calibrated and uses fast correlation based on a matching window centred on the pixel of interest. Specifically, it uses the sum of absolute differences (SAD). It is a suitable off-the-shelf candidate for automotive research.

$$SAD(d) = \sum_{(x,y) \in window} |I_R(x,y) - I_L(x+d,y)| \tag{5.1}$$

Figure 5.1 Left and right images from Bumblebee2 sensor

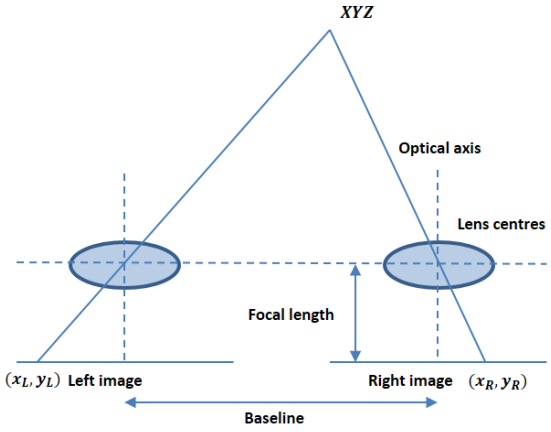


Figure 5.2 Epipolar geometry

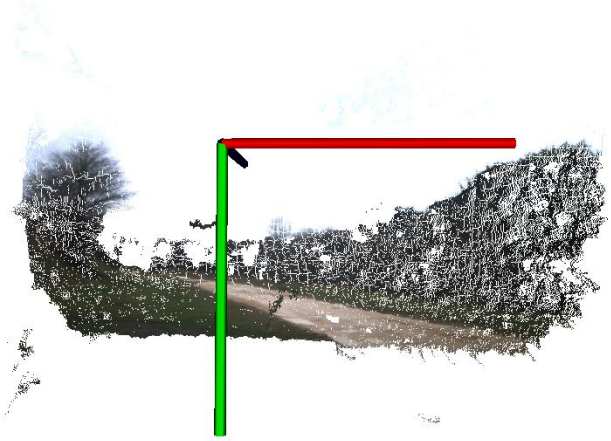


Figure 5.3 Reconstructed point cloud example

Assuming that point XYZ is projected in both left and right images at locations (x_L, y_L) and (x_R, y_R) the world coordinates can be estimated using similar triangles. Let b denote the base-line and f the focal length of the two cameras where the following holds true.

$$\frac{x_L}{f} = \frac{X + b/2}{Z} \tag{5.2}$$

$$\frac{x_R}{f} = \frac{X - b/2}{Z} \tag{5.3}$$

$$\frac{y_L}{f} = \frac{y_R}{f} = \frac{Y}{Z} \tag{5.4}$$

The origin of the world coordinate system is located on the baseline half way between the two lens centres. Previous quantities can be manipulated to obtain expressions for three-dimensional reconstruction of the scene point.

$$X = \frac{b(x_L + x_R)/2}{x_L - x_R} \tag{5.5}$$

$$Y = \frac{b(y_L + y_R)/2}{x_L - x_R} \tag{5.6}$$

$$Z = \frac{bf}{x_L - x_R} \tag{5.7}$$

Where denominator $x_L - x_R$ is the disparity (previously denoted by d) and is computed as a difference between a seed location in the left image and its corresponding match in the right image. This search takes place on the epipolar line and is usually cast as an optimisation problem. Stereo matching is an active field of research with popular benchmarking data (Scharstein and Szeliski 2002) and methods ranging from local window correlation (Einecke and Eggert 2014) to global methods, incorporating smoothness (i.e. on Markov random field lattice) (Sun, Zheng and Shum 2003) as well as methods that try to match popular descriptors (e.g. SIFT, daisy, edges) (Tola, Lepetit and Fua 2010). As there can be multiple matches for ambiguous structures, a lot of effort goes into defining an appropriate optimisation function or energy that best reflects prior knowledge about the problem domain. Often times, the most accurate disparity is estimated by finding an approximate solution of the energy function. However such methods are prohibitively expensive to compute and that renders them unusable for visually demanding automotive applications. Three-dimensional reconstruction is a milestone towards environment perception which is of utmost importance for any driving assist. Subsequent feature extraction and evaluation builds on the complexity of the stereo algorithm. Therefore a simple and efficient stereo algorithm is highly desirable. With Bumblebee2 this energy function

is cast as a sum of absolute pixel differences and the solution is exact. Although in practice there is a lot of room for improving disparity estimation, the stereo camera offers a good balance between speed and accuracy.

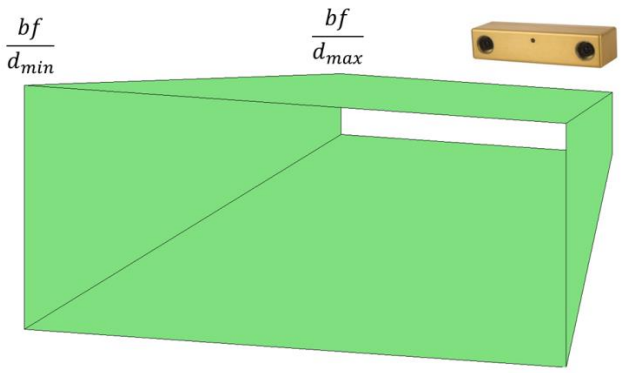


Figure 5.4 Reconstruction volume can be adjusted by restricting disparity search

Points of same corresponding disparity value are coplanar. The reconstruction space is composed of as many parallel planes as there are disparities. Disparity map is typically interpolated to obtain sub-pixel values towards a better discretization. The range field (Figure 5.4) can be constrained by selection of a disparity interval $[d_{min}, d_{max}]$. Far away points are not informative of the class they are representing and are best discarded. Restricting the range field speeds up computation and limits spurious point reconstruction. In a forward driving scenario the road scene contains points for which pixel correspondence is ambiguous. Sky, pothole reflections of sky and points approaching the vanishing line are virtually impossible to reconstruct (Figure 5.5). Moreover, the error of range information obtained via stereo grows quadratically with the distance (Bajracharya *et al.* 2008).

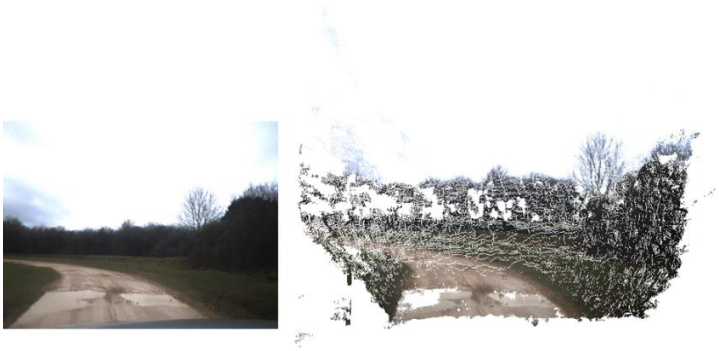
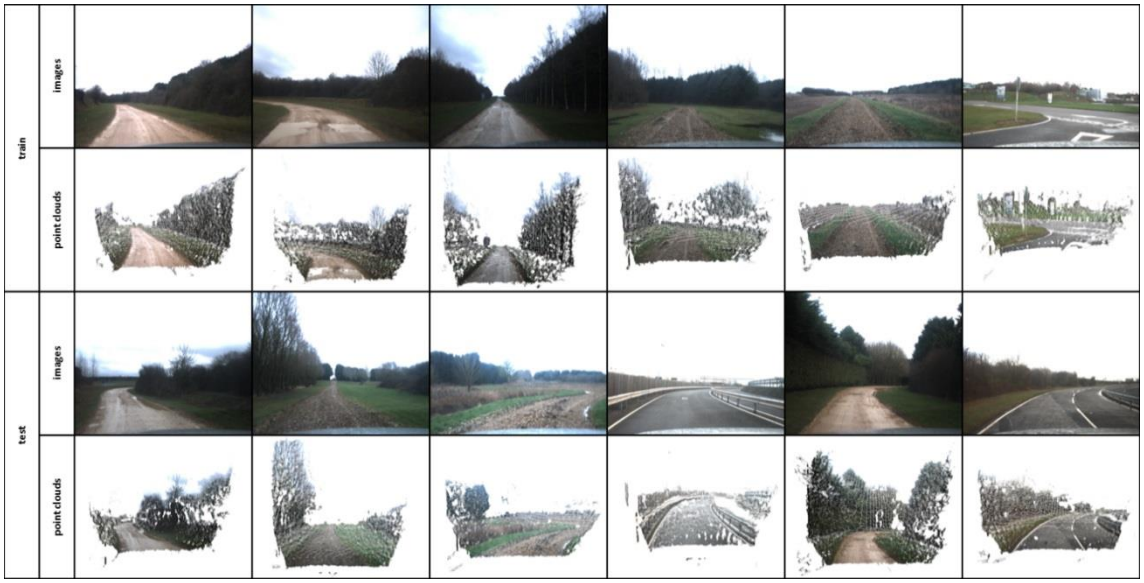


Figure 5.5 Stereo reconstruction and potholes

5.4 Experiments

Table 5.1 Examples of images with corresponding point clouds used for training and testing



Experiments of this chapter use the same setup, dataset and colour coding conventions as outlined in Chapter 4. The original JLR dataset was recorded with a colour stereo camera hence every frame that was previously used is merely the left reference frame of a stereo pair. More specifically, it is still the frame contributing with visual appearance features towards every atomic unit used for evaluation. In effect, the train, validation and test sets (Table 5.1) now have corresponding point clouds. For diagrams showing an overview of the 2D (i.e. image plane) and 3D (i.e. point cloud) processing steps that a typical image undergoes in pursuit of semantics, the reader is referred to Appendix A, diagrams A.3 for scalar prediction and A.4 for

structured prediction. Moreover, the reader is referred to Appendix B code snippet B.6 for code related to related to feature extraction from point clouds using Point Cloud library (http://pointclouds.org/) in conjunction with Triclops application program interface. The latter is used at an early stage to reconstruct point clouds from left and right image sequences and to manipulate the Bumblebee2 stereo camera (https://www.ptgrey.com/triclops).

5.4.1 Features

Two types of features are computed for superpixels (or equivalently supervoxels): node and edge features (Figure 5.6). Node features are extracted from each superpixel individually and therefore can be utilised by all discriminative learning algorithms (i.e. SVM, RF, LR and CRF). Conversely, the edge features are extracted between adjacent pairs of superpixels making them exclusively tailored for a structured output prediction framework (Figure 5.7).

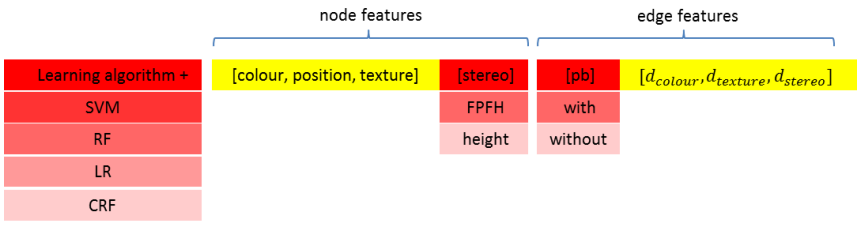


Figure 5.6 Overview of the configurable models. Algorithms and features (seen in red boxes) are explored while the other components (seen in yellow boxes) remain fixed. Stereo surface saliency can be toggled to use FPFH or height. Edge features only apply to structured prediction (i.e. CRF) and the probability of boundary feature can be used or discarded

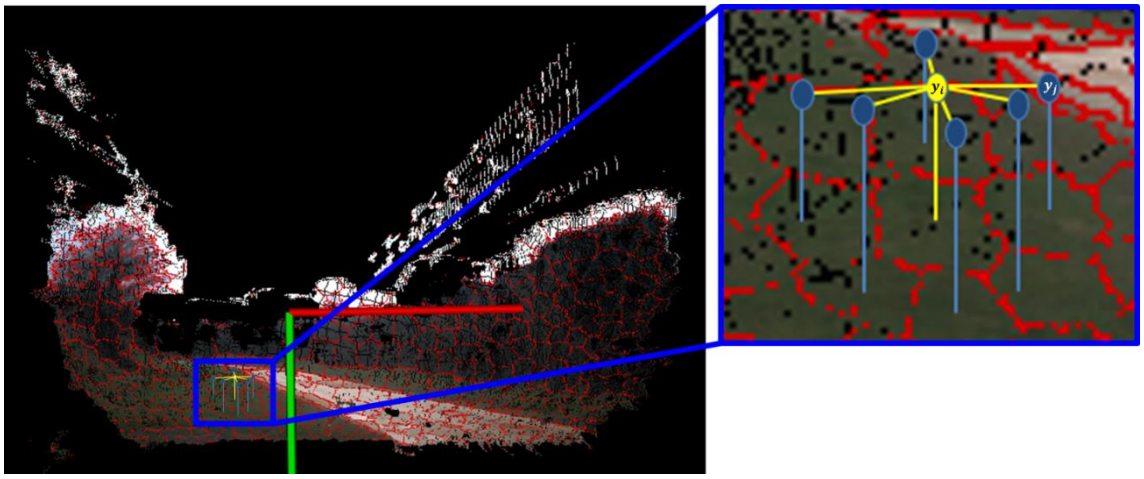


Figure 5.7 Atomic units are pseudo-supervoxels and smooth labelling is encouraged using structured output learning with CRF

5.4.1.1 Node features

- Colour, position, texture

  The part of the superpixel node feature $[colour, position, texture]$ accounting for colour, texture and position is kept unchanged from the previous experiments as follows: colour is summarised using 64 Lab prototypes, superpixel location is distributed across 64 image positions and texture is summarised using 300 daisy prototypes (daisy 2).

- 3D information

  The real world information generated by stereo has been exploited as part of the full node (or superpixel) feature $[colour, position, texture, stereo]$ in two different ways: in one approach, a superpixel feature is generated using a 3D descriptor, in the other simply using the raw height coordinate of the point cloud. In both cases the superpixel pattern is described as a histogram of prototypes. Features computed using the real world domain are summarised using a bag of spatial words as opposed to visual. To allow feasible computation time for 3D descriptors the point cloud is typically downsampled. The point cloud generated by the stereo camera is downsampled using a voxel grid. Voxel grid downsampling works by partitioning the space into 3D boxes

and approximating each box by the centroid of the points it encompasses. This procedure not only reduces the number of points but has also the advantage of filtering the initial noisy stereo reconstruction. If 3D descriptors are to be informative of the surface they represent they should capture the true underlying properties of their class and not be influenced by the inherent stereo outliers. After downsampling the point cloud represents the same world space albeit in a more discrete fashion. Points can be reprojected into the left reference image and likewise any 3D descriptors or cloud coordinates can be tied to pixels as a way to augment the available information in the image domain. To give an example, if the image resolution is $m \times n$ and the descriptor has p dimensions then the end data structure is represented as $m \times n \times p$. Similarly, if height is used the data structure becomes $m \times n \times 1$. Stereo is not able to reconstruct all points that are present as pixels in the reference image so a one to one correspondence cannot be established between pixels and 3D descriptors or height coordinates. It might be due to the well-known stereo disadvantages of non-textured or ambiguous regions. In addition to that, given the length of the baseline, some parts of an object may not be visible in both cameras altogether. In that case certain patches will appear as shadows in the disparity map. These shortcomings are overcome by initialising the data structure with a default value and then augmenting with only those descriptors or height coordinates for which points could be reconstructed and reprojected at the same location where they came from (Figure 5.8).

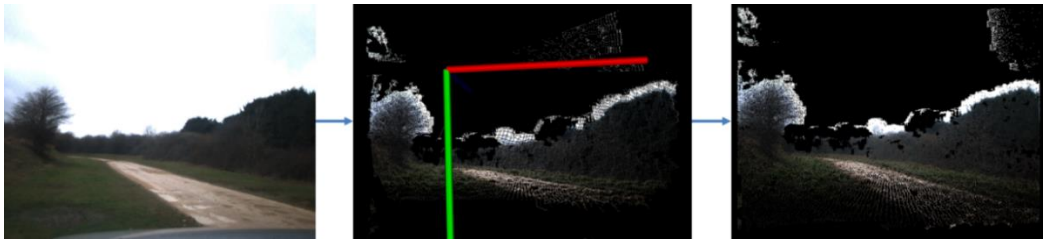


Figure 5.8 Left reference image pixels can be reconstructed into real world coordinates given the valid disparity values. The reconstructed space and features thereafter can be used to enhance pixel information back in the image domain

Normal estimation

Once the point cloud has been downsampled, another important milestone in the computation of most 3D descriptors is normal estimation (Figure 5.9). Estimating the normal of each point within a cloud requires that points around it be taken into account. One of the simplest methods to find point normals is to analyse the eigenvalues and eigenvectors of the covariance matrix that summarizes the spread of points within a neighbourhood. This is equivalent to finding the normal of a plane that is tangent to the point cloud surface. Plane fitting with a cost function such as least squares can be solved using principle component analysis. The covariance matrix is computed for a neighbourhood of points that are encapsulated within a sphere of predefined size depending on how much support is actually needed for normal estimation. Assuming that k points are present within the sphere which is centred at point p_i then the covariance matrix becomes:

$$Cov(p_i) = \frac{1}{k}\sum_{i=1}^{k}(p_i - \overline{p})(p_i - \overline{p})^T, \quad \overline{p} = \frac{1}{k}\sum_{i=1}^{k} x_i, y_i, z_i \tag{5.8}$$

$$Cov \cdot \overrightarrow{v_j} = \lambda_j \cdot \overrightarrow{v_j}, \quad j \in \{0, 1, 2\} \tag{5.9}$$

The normal at point p_i becomes one of the covariance eigenvectors $\overrightarrow{v_j}$, specifically the eigenvector that has the smallest corresponding eigenvalue λ_j.

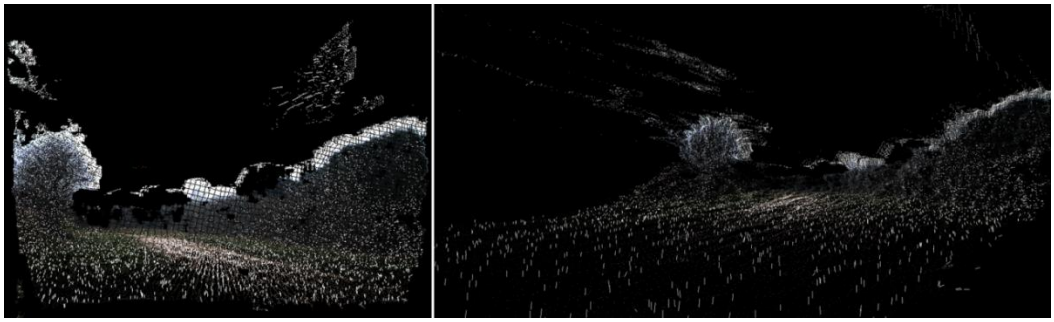


Figure 5.9 Point cloud normal estimation

Fast point feature histograms (FPFH)

Fast point feature histograms (Rusu, Blodow and Beetz 2009) are pose invariant 3D descriptors computed for each point in the cloud. They capture surface saliency by means of relative angular variation between the point normals of a predefined neighbourhood. Although somewhat misleading due to being referred to as histograms these descriptors are not counting occurrences across prototypes (in the bag-of-words sense) but across value ranges. FPFH are an improved version of the previous point feature histograms (PFH) descriptors (Rusu *et al.* 2008a), (Rusu *et al.* 2008b). Just as the previous descriptors, FPFH requires the estimation of angular variations between certain pairs of point normals in a neighbourhood. This is achieved by defining a Darboux frame (Figure 5.10) for every selected pair of points p_i and p_j with corresponding normals n_i and n_j. Assuming that the normal n_i is making a smaller angle with the line joining the points than n_j, the Darboux frame becomes $(u = n_i, v = (p_j - p_i) \times u, w = u \times v)$. Angular variations of the pair n_i, n_j can be expressed subsequently:

$$\alpha = v \cdot n_j \tag{5.10}$$

$$\phi = (u \cdot (p_j - p_i))/\|p_j - p_i\|_2 \tag{5.11}$$

$$\theta = \arctan(w \cdot n_j, u \cdot n_j) \tag{5.12}$$

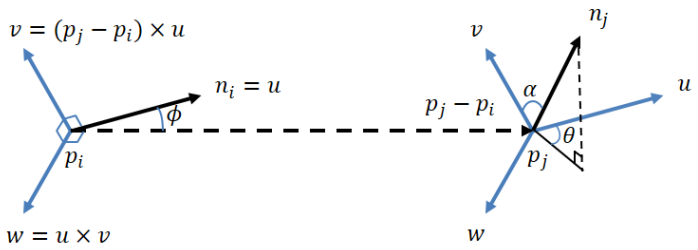


Figure 5.10 Point normals in a Darboux frame

The original descriptors contained a fourth dimension characterising the Euclidean distance between points in addition to the angular variations. This has been removed altogether without affecting robustness since in most cases points that are far away from the view point are further apart from each other (Rusu, Blodow and Beetz 2009). Previous formalisms are common to both FPFH and PFH, however they differ in the manner they leverage point pairs within the proximity of a seed towards the final descriptor. The original PFH computes α, ϕ, θ for every possible pair of points within a sphere of predefined radius centred on a seed point. This results in a theoretical complexity of $O(nk^2)$ for a cloud of n points each having k neighbours. In spite of its discriminative power reported by the literature, processing times were a major drawback (Rusu *et al.* 2008a), (Rusu *et al.* 2008b). Adjustments were made to the original PFH that resulted in a faster descriptor (i.e. FPFH) as follows. First a simplified point feature histogram (SPFH) is computed for all points. This is done by considering only the point pairs formed between a seed point and its k neighbours. Then for each point the SPFH is merged with a weighted SPFH average of its k neighbours within a sphere to create the FPFH descriptors. If the sphere has a specified radius r then points as far as $2r$ can

contribute towards the estimation of FPFH (Figure 5.11). In effect this is a way to re-capture some of the pairs or connections that were present in the original PFH descriptor. The weighting parameter ω_j considers a distance measure between the seed and its neighbours essentially making close points matter more than those further. This has been experimentally proved to be just as effective whilst having a reduced theoretical computational complexity of $O(nk)$.

$$FPFH(p_i) = SPFH(p_i) + \frac{1}{k}\sum_{j=1}^{k}\frac{1}{\omega_j}\cdot SPFH(p_j) \tag{5.13}$$

$$\omega_j = \|p_i - p_j\|_2 \tag{5.14}$$

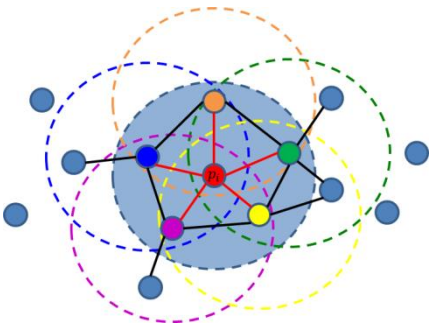


Figure 5.11 Point neighbourhoods are captured in circles (spheres in 3D) towards the computation of FPFH at point p_i

Each of the three angular variations α, ϕ, θ is binned using 11 subdivisions. Individual histograms are concatenated resulting in a final descriptor with 33 dimensions for each point in the cloud. Similar to their 2D descriptor counterpart, a fraction of the training point clouds with their corresponding 3D descriptors are set aside. Clustering using the same probabilistic variant of K-means as before learns 300 prototypes. Subsequently the surface saliency within a supervoxel (or augmented superpixel) can be summarised as a bin count across those prototypes. This approach stems entirely from

the classical bag of words albeit from a research venue consecrated to indoor environments and RGB-D acquiring sensors (Hernandez-Vela *et al.* 2012).

Height

Height is one of the real world dimensions and is densely estimated. Representation of height information as a normalised histogram is robust to outliers even without downsampling the point cloud. Most height coordinates would be close to their real world value with occasional outliers sprinkled across the supervoxel. Sky does not have height information and that makes it the most likely class to keep the default descriptor value. Ideally if height is to be used as a descriptor, left and right camera coinciding planes should be perfectly perpendicular to the horizontal real world (i.e. road plane) in which objects lie. This is to ensure that an object's height does not vary with distance and it is a rather intrinsic property of the class. For example, the real world height of a shrub should remain constant or within its typical tolerance as the vehicle approaches or departs from it. The standard procedure is to rotate and translate the point cloud. This however is not practical for a moving vehicle for a number of reasons. Road topography might change drastically from a region to another. Since the stereo rig is fitted onto a vehicle, the camera plane is subject to road contact via suspensions which renders rotation and translation parameters unusable from one frame to another. Traditionally to overcome this problem parameters are estimated at each frame by matching key-points within successive point clouds. Daunting processing times as well as having the emphasis on terrain types, within a likely off-road scenario as opposed to objects, only exacerbate the need to avoid cloud transformations. Rather than getting a narrow range of height distribution across a class, a broader but still informative range is preferred. To achieve this, the stereo camera should only be lightly tilted with respect to a typical flat road and disparity search should be restricted to

limit cloud reconstruction. Minimum disparity d_{min} controls the maximum reconstruction distance. Partitioning the space in such a manner makes height not only consistent in its range for each class but also more reliable. Recall that with stereo the reconstruction accuracy drops for distant points. From a number of such training point clouds K-means clustering with probabilistic seeding is applied to discover 64 height prototypes. Subsequently each superpixel is described by a height histogram across those prototypes.

5.4.1.2 Edge features

Edge features are computed between neighbouring superpixels. The same formalisms that were previously applied to generate edge features can now be extended to incorporate real world measurements. With the exception of the 3D information from stereo for which an extra dimension is introduced in the edge features, the other dimensions are brought forward from the 2D realm.

- Probability of boundary, colour, texture

  The first three dimensions of the edge features are kept as the best performing trio $[pb, d_{colour}, d_{texture}]$ in the left reference image experiment described in chapter 4. The probability of boundary can now be interpreted as both a measure of superpixel adjacency as well as a measure of supervoxel adjacency. Because image boundaries are not necessarily real world boundaries, experiments are carried out both $with$ and $without$ the probability of boundary as part of the edge feature. The discrete choice aforementioned is also motivated by the huge computation overhaul of such a feature even if it were to be estimated using the real world point cloud. Colour dissimilarity is again computed as the Euclidean distance between the average Lab values of neighbouring superpixels. Only the best texture configuration from previous experiments is

carried forward. Namely the texture dissimilarity is measured as a chi-squared distance between histograms of 300 daisy prototypes, previously denoted by $daisy\ 2$.

- 3D information

  The complete edge feature is of the form $[pb][d_{colour}, d_{texture,}\ d_{stereo}]$ where d_{stereo} denotes a distance measure between the statistics of neighbouring super-voxels. Again these statistics are summarised by histograms of descriptor prototypes obtained from the point clouds. The intersection kernel (Barla, Odone and Verri 2003) has been found to be the most appropriate distance measure among a handful of other candidates such as correlation, chi-squared and Bhattacharyya distance. The selection of a metric was guided by the performance that could be leveraged experimentally.

$$d_{stereo}(h_1, h_2) = \sum_{i=1}^{p} \min(h_1(i), h_2(i)), \quad p = \begin{cases} 300\ FPFH \\ 64\ height \end{cases} \tag{5.15}$$

5.4.2 Evaluation

The JLR dataset, described previously in Chapter 4, has been utilized throughout these experiments for evaluating the quality of semantic segmentation. The ground truth for superpixels in image domain has remained the same. The most notable difference is that now there is additional saliency coming from stereo reconstruction. In order to evaluate the performance of scalar and structured output learning in conjunction with flavours of surface saliency from a ranging sensor (e.g. stereo camera) the same measures (Sokolova and Lapalme 2009) have been used as in Chapter 4, namely overall superpixel accuracy, confusion matrix, error reduction, recall and intersection-over-union. The baseline and the top performing model of Chapter 4 are replicated here (for comparison) in order to establish if stereo does indeed lead to more

accurate semantic segmentation of terrain types and what is a good learning scheme to achieve that. The following tables allow for a model comparison of per-class accuracies expressed as recall (Table 5.2) and intersection-over-union (Table 5.3) as well as their average across all classes. Entries are sorted according to overall accuracies and error reduction of each model with respect to a baseline. Just as suggested in Chapter 4, the intersection-over-union is a better suited per class accuracy measure since it penalises both under-estimation and over-estimation. Same labelling outside the natural borders of a certain class should be penalised, from the perspective of an accuracy scoring criteria, even if the labelling within the borders is accurate. Therefore, Table 5.3 should be considered as primary guideline when selecting a particular model to predict semantics across a range of classes. Looking at intersection-over-union outside the 2D band, classes such as {dirt, gravel, shrubs, tarmac} show large performance gains with minor improvements for {grass}, while {tree, sky} compare favourably with the top performing configurations. The accuracies across some classes may appear to be similar regardless of the used model. This suggests that accuracy is due to texture and stereo does not help discriminate even further. The best overall accuracy across the test set is obtained using structured prediction with texture and stereo. Texture saliency is obtained with bag-of-visual-words using a vocabulary of 300 daisy prototypes while surface saliency is obtained with bag-of-spatial-words using a vocabulary of 64 height values. The structured learning framework appears to alleviate the need to use probability of boundary when stereo information is present. In particular the surface saliency variant based on simple height is both fast and discriminative, leading to improvements in the quality of semantic segmentation. Confusion matrices (Figure 5.12 and Figure 5.13) clarify the gains even further and colour coded results can be visually inspected in Table 5.4 and Table 5.5.

Table 5.2 Recall accuracies; models are sorted in ascending order of their overall accuracy: (a) RF + [daisy 2, FPFH], (b) SVM + [daisy 2, FPFH], (c) LR + [daisy 2, FPFH], (d) LR + [daisy 2, height], (e) SVM + [daisy 2, height], (f) RF + [daisy 2, height], (g) SVM + [textons], (h) CRF + [daisy 2] + [pb], (i) CRF + [daisy 2, FPFH] + [pb], (j) CRF + [daisy 2, FPFH], (k) CRF + [daisy 2, height] + [pb], (l) CRF + [daisy 2, height]

	model	overall acc	E(model)	grass	tree	sky	dirt	gravel	shrubs	tarmac	void	average recall
3D	(a)	88.10	-7.89	93.98	90.36	99.31	35.18	81.57	36.61	72.22	75.07	73.04
	(b)	88.12	-7.71	92.66	88.37	99.18	40.44	80.18	46.62	70.24	76.26	74.24
	(c)	88.18	-7.16	92.20	86.40	**99.34**	37.74	80.90	47.49	75.06	76.00	74.39
	(d)	88.22	-6.80	91.45	86.54	99.32	39.95	81.12	44.69	76.66	77.11	74.61
	(e)	88.40	-5.17	91.31	88.38	99.18	43.09	79.79	46.18	73.62	79.04	75.07
	(f)	88.65	-2.90	93.86	90.29	99.30	36.98	83.36	33.85	79.98	76.27	74.24
2D	(g)	88.97	0	93.09	90.26	99.29	47.19	82.15	48.18	68.20	**79.22**	75.95
	(h)	90.56	14.42	92.56	90.25	99.22	59.74	84.67	50.04	88.81	77.39	80.34
3D	(i)	90.69	15.59	93.80	90.11	99.26	54.79	84.18	50.71	94.66	74.46	80.25
	(j)	90.73	15.96	**94.15**	90.20	99.26	55.15	**84.73**	47.91	**94.84**	75.55	80.23
	(k)	90.81	16.68	92.21	89.27	99.26	**60.82**	84.59	**52.36**	93.56	77.17	**81.15**
	(l)	**90.92**	**17.68**	93.18	**90.75**	99.23	60.19	84.72	51.57	93.66	75.22	81.06

Table 5.3 Intersection/union accuracies; models are sorted in ascending order of their overall accuracy: (a) RF + [daisy 2, FPFH], (b) SVM + [daisy 2, FPFH], (c) LR + [daisy 2, FPFH], (d) LR + [daisy 2, height], (e) SVM + [daisy 2, height], (f) RF + [daisy 2, height], (g) SVM + [textons], (h) CRF + [daisy 2] + [pb], (i) CRF + [daisy 2, FPFH] + [pb], (j) CRF + [daisy 2, FPFH], (k) CRF + [daisy 2, height] + [pb], (l) CRF + [daisy 2, height]

	model	overall acc	E(model)	grass	tree	sky	dirt	gravel	shrubs	tarmac	void	average ∩/∪
3D	(a)	88.10	-7.89	79.08	73.70	98.17	31.03	67.79	27.56	55.47	66.04	62.36
	(b)	88.12	-7.71	80.83	75.55	96.07	34.84	64.71	33.39	60.52	64.86	63.85
	(c)	88.18	-7.16	**82.03**	74.87	97.61	32.18	63.77	33.72	62.54	60.82	63.44
	(d)	88.22	-6.80	81.51	74.38	97.68	35.53	63.81	32.55	63.86	60.63	63.74
	(e)	88.40	-5.17	80.40	75.80	96.01	37.99	65.69	33.63	63.05	65.68	64.78
	(f)	88.65	-2.90	79.64	73.85	98.18	33.62	67.02	27.00	63.93	**66.99**	63.78
2D	(g)	88.97	0	82.01	**77.80**	**98.45**	32.16	67.27	33.20	62.28	66.55	64.96
	(h)	90.56	14.42	81.68	77.23	98.25	53.51	71.85	38.39	78.78	65.27	70.62
3D	(i)	90.69	15.59	81.68	77.40	98.22	50.75	72.34	39.88	78.04	65.77	70.51
	(j)	90.73	15.96	81.61	77.45	98.12	50.72	72.44	38.48	79.46	66.32	70.58
	(k)	90.81	16.68	81.51	76.79	98.23	**56.62**	**72.85**	**40.36**	**81.13**	65.49	71.62
	(l)	**90.92**	**17.68**	81.54	77.37	98.27	56.36	72.66	40.10	80.61	66.72	**71.70**

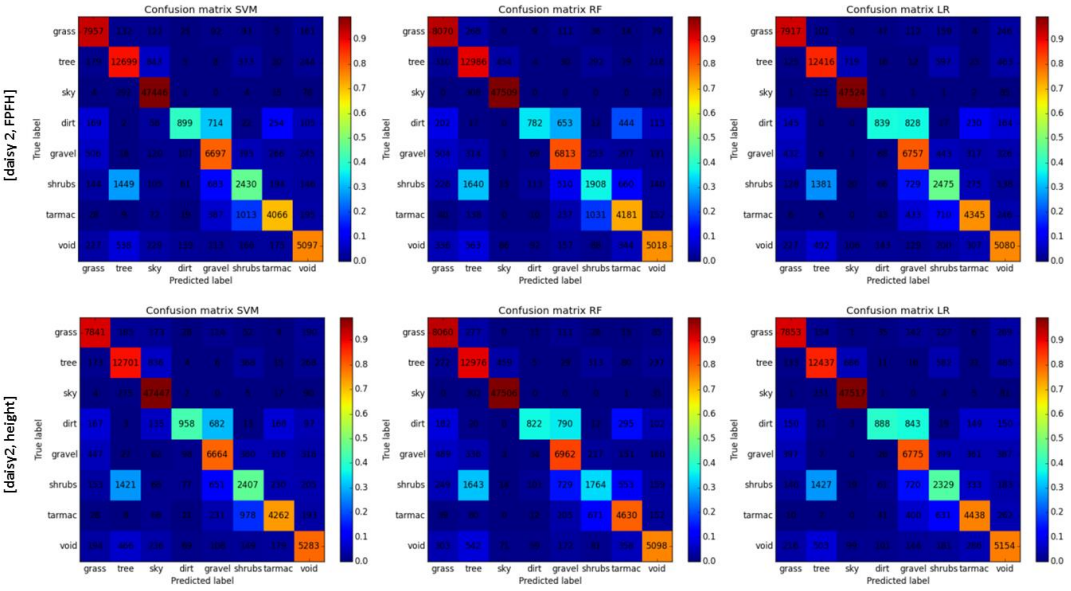


Figure 5.12 Confusion matrices of scalar predictors with surface saliency flavours

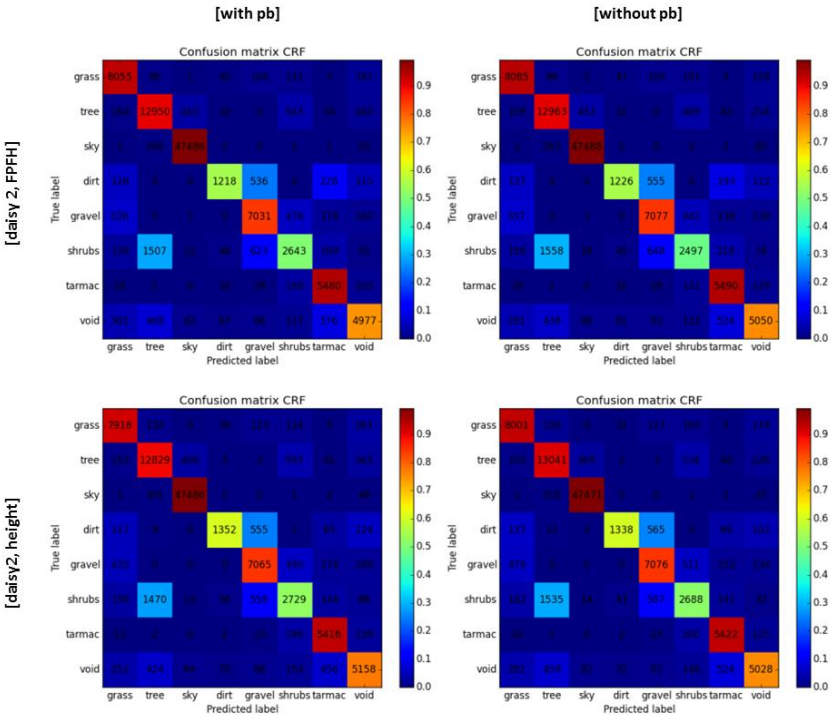


Figure 5.13 Confusion matrices of structured output learning with surface saliency flavours, using and discarding probability of boundary

Table 5.4 Colour coded results. From left to right: input image, RF + [daisy 2, height], SVM + [textons], CRF + [daisy 2] + [pb], CRF + [daisy 2, FPFH] + [pb], CRF + [daisy 2, FPFH], CRF + [daisy 2, height] + [pb], CRF + [daisy 2, height]

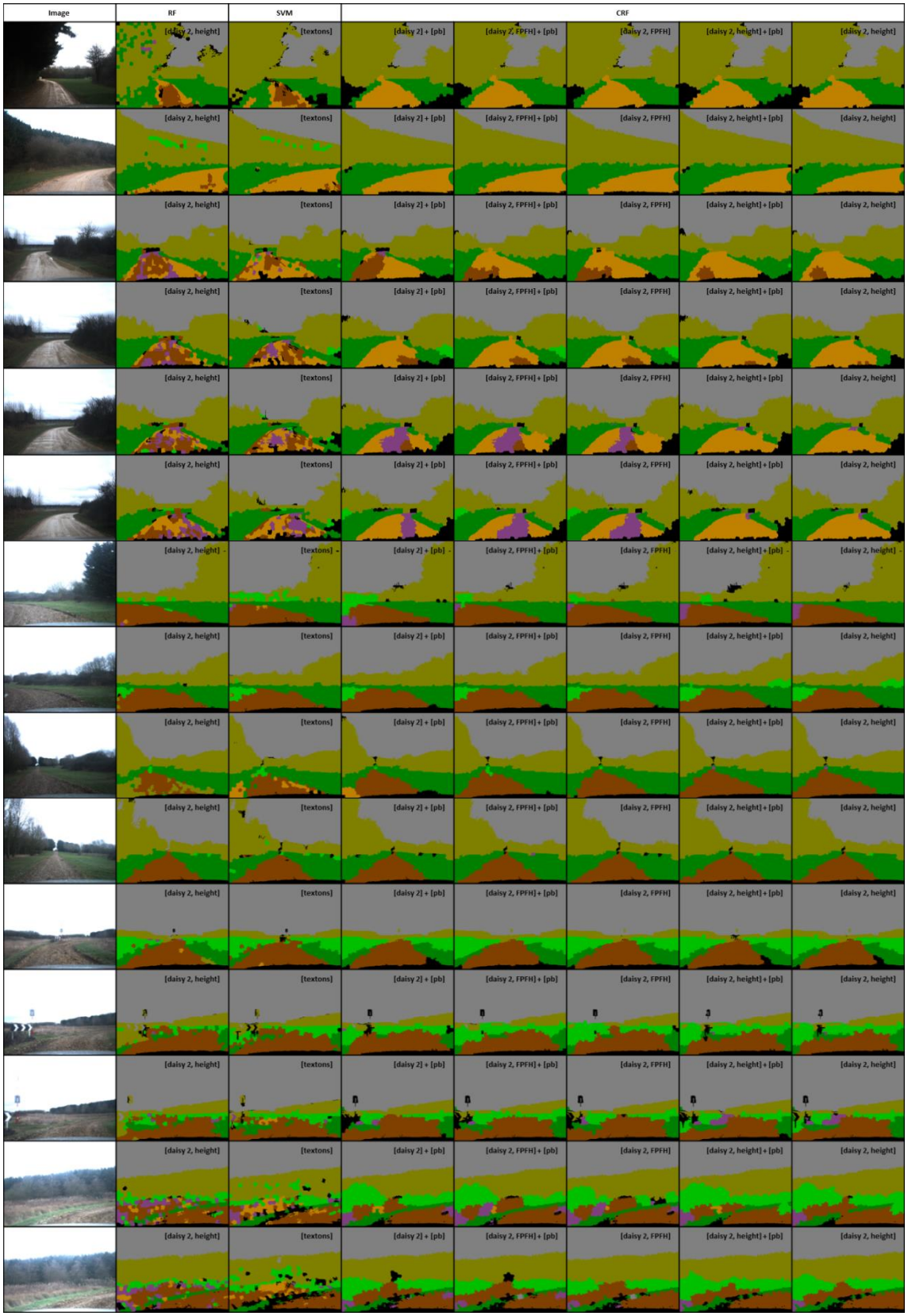
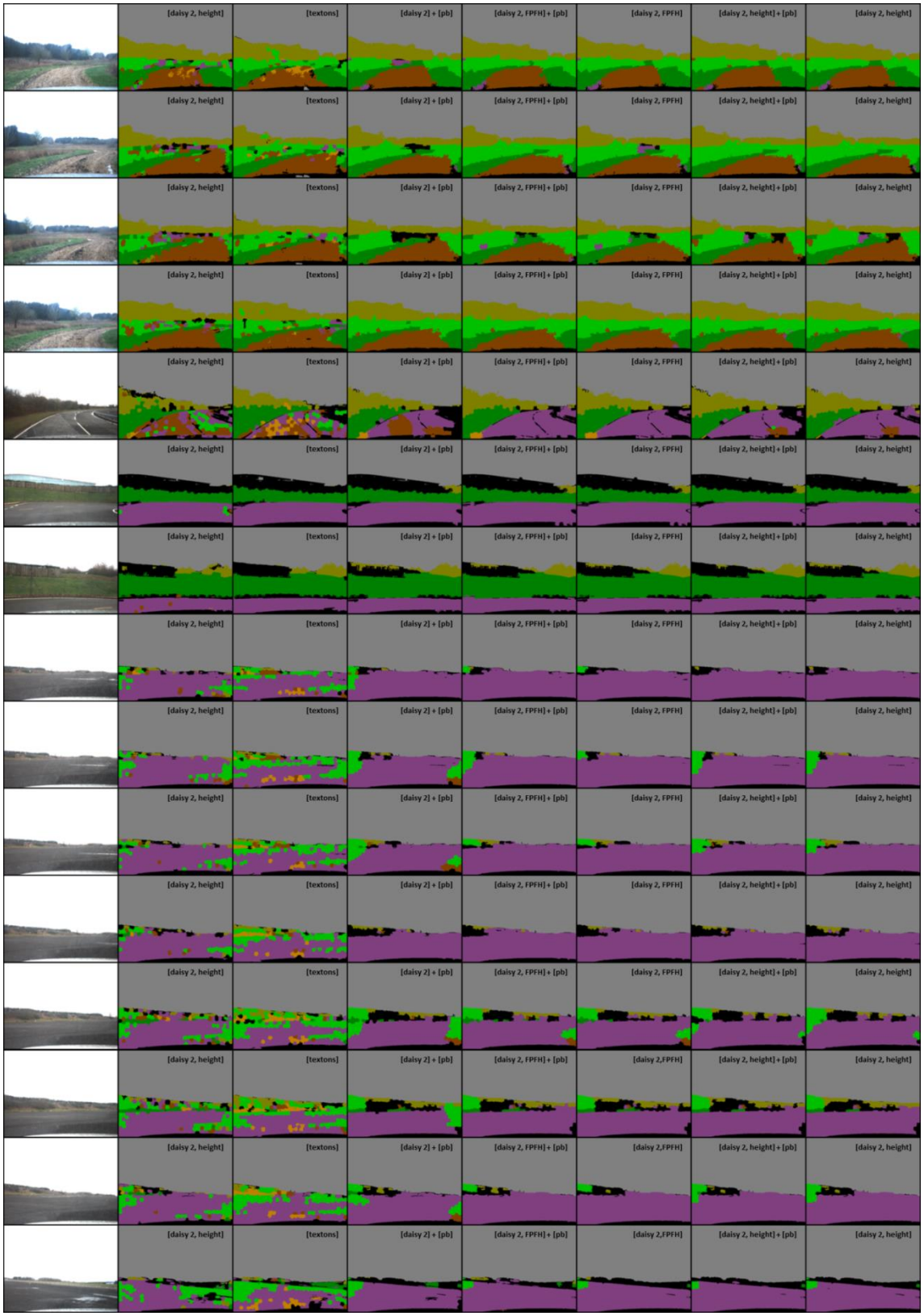


Table 5.5 Colour coded results (continued). From left to right: input image, RF + [daisy 2, height], SVM + [textons], CRF + [daisy 2] + [pb], CRF + [daisy 2, FPFH] + [pb], CRF + [daisy 2, FPFH], CRF + [daisy 2, height] + [pb], CRF + [daisy 2, height]



5.4.2.1 Processing time

Experiments have been run on the same machine as the setup described in Chapter 4. Again various processing steps are timed in order to establish the computational expenditure of various milestones along the path of prediction (Table 5.6). As expected, capturing surface saliency from stereo vision using one of the raw point cloud coordinates (i.e. height) is the cheaper alternative. Structured output learning configured with daisy 2 and height bag-of-features achieves not only the best overall superpixel accuracy but also competitive computation time (Figure 5.14) among the tested learning schemes.

Table 5.6 Processing times in s; models are sorted in ascending order of their overall accuracy: (a) RF + [daisy 2, FPFH], (b) SVM + [daisy 2, FPFH], (c) LR + [daisy 2, FPFH], (d) LR + [daisy 2, height], (e) SVM + [daisy 2, height], (f) RF + [daisy 2, height], (g) SVM + [textons], (h) CRF + [daisy 2] + [pb], (i) CRF + [daisy 2, FPFH] + [pb], (j) CRF + [daisy 2, FPFH], (k) CRF + [daisy 2, height] + [pb], (l) CRF + [daisy 2, height]

	model	feature dimension		superpixels per frame (on average)	features extraction per frame (on average)								features evaluation per frame (Matlab/Python)	total time per frame (640×480)
					all nodes				all edges					
		node	edge		colour (C++)	position (C++)	texture (Matlab)	stereo (C++)	pb (Matlab)	d_{colour} (C++)	$d_{texture}$ (C++)	d_{stereo} (C++)		
3D	(a)	728	n/a	768	0.164	0.001	2.806	3.620	n/a	n/a	n/a	n/a	0.097	6.688
	(b)	728	n/a	768	0.164	0.001	2.806	3.620	n/a	n/a	n/a	n/a	27.262	33.853
	(c)	728	n/a	768	0.164	0.001	2.806	3.620	n/a	n/a	n/a	n/a	0.001	6.592
	(d)	492	n/a	768	0.164	0.001	2.806	0.258	n/a	n/a	n/a	n/a	0.001	3.230
	(e)	492	n/a	768	0.164	0.001	2.806	0.258	n/a	n/a	n/a	n/a	17.384	20.613
	(f)	492	n/a	768	0.164	0.001	2.806	0.258	n/a	n/a	n/a	n/a	0.076	3.305
2D	(g)	192	n/a	768	0.164	0.001	1.364	n/a	n/a	n/a	n/a	n/a	5.781	7.310
	(h)	428	3	768	0.164	0.001	2.806	n/a	60.658	0.002	0.011	n/a	0.888	64.530
3D	(i)	728	4	768	0.164	0.001	2.806	3.620	60.658	0.002	0.011	0.002	0.848	68.112
	(j)	728	3	768	0.164	0.001	2.806	3.620	n/a	0.002	0.011	0.002	0.925	7.531
	(k)	492	4	768	0.164	0.001	2.806	0.258	60.658	0.002	0.011	0.001	0.857	64.758
	(l)	492	3	768	0.164	0.001	2.806	0.258	n/a	0.002	0.011	0.001	0.844	4.087

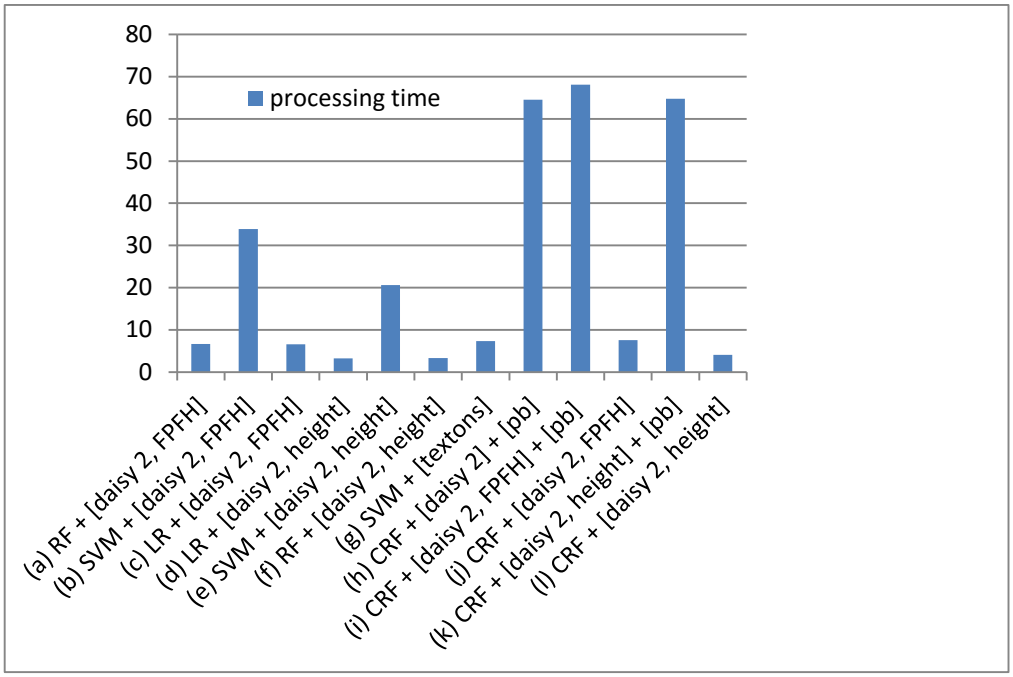


Figure 5.14 CRF + [daisy 2, height] is competitive in terms of processing time and achieves best accuracy on the JLR Dataset. Computation times are representative for a JLR frame of 640 x 480 pixels. Such times should be regarded in relative terms since both the image resolution and the test platform may change

5.4.3 Comparison to prior work

Table 5.7 Annotated classes present in the proprietary JLR dataset vs. the public KITTI dataset

dataset	grass	tree	sky	dirt	gravel	shrubs	tarmac superclass	road subclass	void
JLR	X	X	X	X	X	X	X		X
KITTI								X	X

Comparison to previous work has been done using two different datasets, namely the proprietary JLR dataset and the public KITTI dataset (Table 5.7). Typical image samples corresponding to the two datasets can be visually inspected in Appendix C. Firstly on the JLR dataset a number of algorithms and features have been tested and compared with a baseline made up of an SVM classifier and bag-of-visual-words very similar to the work of Filitchkin and Byl (2012). From a scenario perspective this is an appealing and critical comparison since the method described by Filitchkin and Byl (2012) is tailored for the classification of various terrain classes

(i.e. {tarmac, grass, gravel, mud, soil and woodchips}). Secondly, the best performing method on the JLR dataset has been benchmarked on the challenging KITTI urban road dataset albeit by only considering classification of two classes: tarmac and void (i.e. road and non-road, according to the KITTI class naming convention). Comparison to KITTI is motivated by both the need to prove method versatility from a similar ADAS standpoint as well as by the fact that terrain datasets with all required classes and desired intra-class variance are hard to come by. Existing datasets contain only a subset of the required classes and often lack stereo data. First comparison (i.e. JLR dataset) reveals superior performance of the proposed structured prediction model. The introduction of 3D stereo improves the overall classification accuracy and benefits the majority of classes. The rest only suffer a minor setback and are still performing close to the best reported superpixel intersection-over-union. It is not a surprise that stereo does not help with sky recognition. Perhaps not so obvious is why tree performance is marginally worsened by the introduction of stereo information. Recall that the smoothness assumption (i.e. prior) leverages similarities among superpixel statistics belonging to the same class. Texture and colour are likely to be smooth in image regions depicting trees. While trees are easily reconstructed by the stereo algorithm, the real world statistics attributed to each superpixel may vary greatly within a given neighbourhood. For example, in a tree collection foliage and branches might have different point cloud surfaces and since they are erected from the ground their height varies greatly too. In a second round of comparisons the suggested approach is benchmarked on the state of the art KITTI dataset resulting in improvements over a number of previous approaches. Although pixel labelling results for images can be queried in both perspective and birds eye view (BEV), KITTI comparisons make use of specific BEV pixel accuracy scoring (Fritsch, Kuhnl and Geiger 2013) as this is more relevant to spatial occupancy of the scene. Pixel based metrics include maximum F_1-measure, average precision, precision,

recall, false positive rate and false negative rate. The mathematical formulations of such measures are reproduced here for completeness.

$$false\ positive\ rate = \frac{FP}{FP + TN} \tag{5.16}$$

$$false\ negative\ rate = \frac{FN}{FN + TP} \tag{5.17}$$

$$precision = \frac{TP}{TP + FP} \tag{5.18}$$

$$recall = \frac{TP}{TP + FN} \tag{5.19}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.20}$$

For methods whose outputs are confidence maps instead of binary maps a threshold t is chosen that will give rise to a maximum F_1 measure. In fact the KITTI road benchmark ranks the evaluated methods according to such measure (Fritsch, Kuhnl and Geiger 2013).

$$F_{max} = \max_t F_1 \tag{5.21}$$

The average interpolated precision is expressed as an average of interpolated precisions taken across 11 recall levels r namely {0%, 10%,…, 100%}. This measure is used to summarize the shape of the precision\recall curve.

$$precision_{interp}(r) = \max_{\tilde{r} \geq r} precision(\tilde{r}) \tag{5.22}$$

$$average\ precision = \frac{1}{11} \sum_{r \in 0,0.1,\ldots 1} precision_{interp}(r) \tag{5.23}$$

The KITTI road data set contains 289 training and 290 testing images. After training and testing the proposed method, results have been obtained in the following relevant categories: {urban marked, urban multiple marked lanes, urban unmarked} road. Both CRF + [daisy 2, height] and CRF + [daisy 2, FPFH] methods labelling superpixel atomic units have been benchmarked and publicly ranked on the challenging KITTI road dataset under the acronyms SCRFH and

SCRFFPFH respectively. Accuracy tables with KITTI metrics (Tables 5.8, 5.9 and 5.10) as well as sample visual results (i.e. overlaid image labelling) of the two proposed methods are provided (Tables 5.11, 5.12 and 5.13). Full table results with all benchmark entries can also be inspected online (http://www.cvlibs.net/datasets/kitti/eval_road.php).

Table 5.8 Urban marked road. MaxF: Maximum F1-measure, AP: Average precision, PRE: Precision, REC: Recall, FPR: False Positive Rate, FNR: False Negative Rate

Method	Setting	MaxF	AP	PRE	REC	FPR	FNR
…	…	…	…	…	…	…	…
ARSL-AMI (Passani, Yebes and Bergasa 2014)	monocular	71.97%	61.04%	78.03%	66.79%	8.57%	33.21%
SCRFFPFH	stereo	70.78%	64.76%	83.88%	61.22%	5.36%	38.78%
SCRFH	stereo	69.34%	60.30%	84.47%	58.81%	4.93%	41.19%
ANN (Vitor *et al.* 2013)	stereo	62.83%	46.77%	50.21%	83.91%	37.91%	16.09%

Table 5.9 Urban multiple marked lanes road. MaxF: Maximum F1-measure, AP: Average precision, PRE: Precision, REC: Recall, FPR: False Positive Rate, FNR: False Negative Rate

Method	Setting	MaxF	AP	PRE	REC	FPR	FNR
…	…	…	…	…	…	…	…
SPlane (Einecke and Eggert 2014)	stereo	82.28%	82.83%	76.85%	88.53%	29.32%	11.47%
SCRFH	stereo	82.08%	80.37%	90.87%	74.83%	8.26%	25.17%
SPlane + BL (Einecke and Eggert 2014)	stereo	82.04%	85.56%	75.11%	90.39%	32.93%	9.61%
ANN (Vitor *et al.* 2013)	stereo	80.95%	68.36%	69.95%	96.05%	45.35%	3.95%
SCRFFPFH	stereo	79.75%	80.37%	90.87%	71.06%	7.85%	28.94%
BL (Fritsch, Kuhnl and Geiger 2013)	monocular	76.02%	78.82%	65.71%	90.17%	51.72%	9.83%

Table 5.10 Urban unmarked road. MaxF: Maximum F1-measure, AP: Average precision, PRE: Precision, REC: Recall, FPR: False Positive Rate, FNR: False Negative Rate

Method	Setting	MaxF	AP	PRE	REC	FPR	FNR
…	…	…	…	…	…	…	…
BL (Fritsch, Kuhnl and Geiger 2013)	monocular	69.50%	73.87%	65.87%	73.56%	12.42%	26.44%
SCRFFPFH	stereo	64.97%	55.97%	82.13%	53.74%	3.81%	46.26%
SCRFH	stereo	64.00%	55.99%	82.16%	52.41%	3.71%	47.59%
ANN (Vitor *et al.* 2013)	stereo	54.07%	36.61%	39.28%	86.69%	43.67%	13.31%

Table 5.11 Urban marked visual results of CRF + [daisy 2, FPFH] and CRF + [daisy 2, height]. Red: false negatives, blue: false positives, green: true positives

Table 5.12 Urban multiple marked lanes visual results of CRF + [daisy 2, FPFH] and CRF + [daisy 2, height]. Red: false negatives, blue: false positives, green: true positives

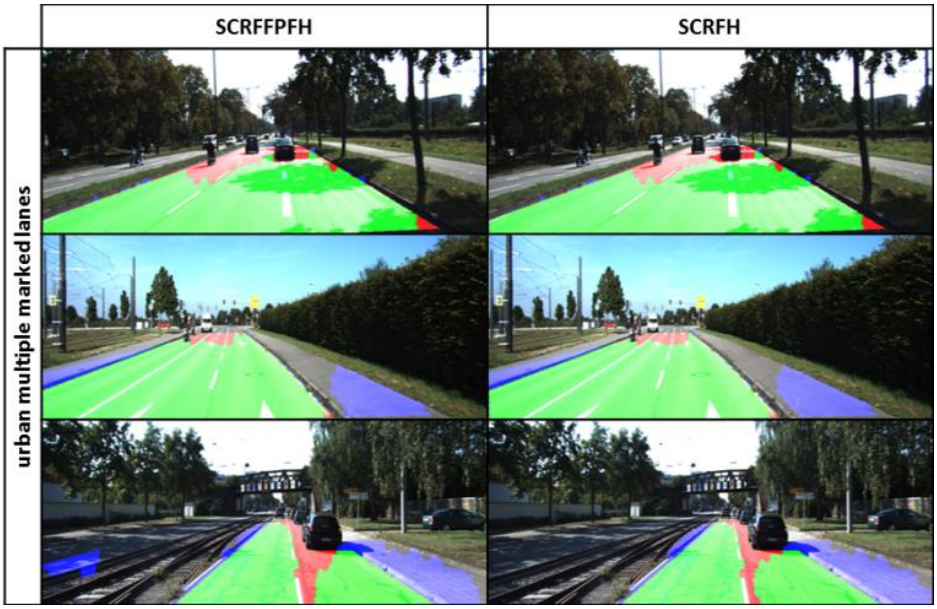
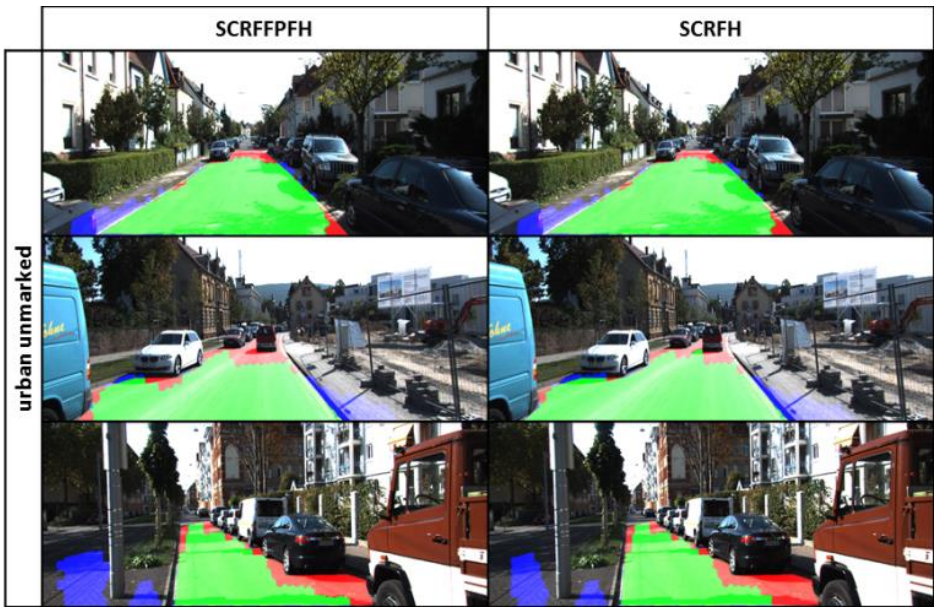


Table 5.13 Urban unmarked visual results of CRF + [daisy 2, FPFH] and CRF + [daisy 2, height]. Red: false negatives, blue: false positives, green: true positives



5.5 Discussion

As the distance increases, it becomes difficult to find support regions (or proximity) for three-dimensional descriptor estimation. Even if these descriptors were to be made adaptive in terms of the local evidence they require, the information contained would be unreliable due to both sparse and noisy distant reconstruction. Height is the most informative and appropriate quantity towards road scene classification with stereo under the given scenario. It is easier to compute than FPFH and easily transformed into a superpixel feature using vector quantisation. It does not require proximity evidence, and is the only basic dimension that changes with the class. In real world coordinates there is a subtle height difference between the similarly textured shrubs and trees but a more obvious difference between grass and trees. Likewise, tarmac appearance can vary but its height tends to be constant. The other two dimensions, namely horizontal and distance displacements are affected by the vehicle position rather than class. The error reduction measure comparing different approaches for classification suggests that the mere introduction of three-dimensional information does not necessarily improve prediction accuracy of amorphous classes, as is the case. Indeed, the image baseline made up of SVM and bag-of-visual-words (Filitchkin and Byl 2012) outperforms the other independent (i.e. scalar) prediction models incorporating real world measures of point clouds. It is only via a structured prediction framework that three-dimensional measures can truly be leveraged. Under this framework segments described by histograms of prototypes are likely to have similar statistics if they belong to the same class. Pairwise terms impose smoothness and affect the final labelling of the CRF. Smoothness has been shown to be a valid assumption for terrain classes not only in the 2D domain but also in 3D. In real world coordinates, distributions across prototypes of height are similar between the segments of the same amorphous class (e.g. grass, dirt, gravel, shrubs and tarmac). This is also the case for distributions across prototypes of local 3D descriptors (e.g. FPFH). They are however invariably more expensive to compute

than height. For example, assigning each frame with FPFH prototypes scales the time it takes to assign it with height prototypes by a factor of approximately 14. Classification of a tree collection does not benefit from the introduction of stereo (as the intersection-over-union measure suggests) but reasonable classification accuracy suggests that smoothness may hold for saliency of visual appearance only. Reduced superpixel size enables individual atomic units to accurately puzzle together to make up the real world, while still reducing the complexity of the graphical model. The suggested over-segmentation technique makes structured prediction possible using long range stereo. Noisy stereo reconstruction is not easily partitioned into locally coherent segments. Rather than performing over-segmentation at the point cloud level, this pre-processing step is done prior to stereo reconstruction. SLIC partitions the left reference image into superpixels or segments. Subsequently superpixels span the subspace of corresponding supervoxels when going from 2D to 3D and vice versa. Novel aspects presented in this chapter include capturing terrain surface saliency using bag of spatial words and in particular FPFH prototypes. The fact that a surface dissimilarity measure can alleviate the need for a visual dissimilarity measure as part of a CRF framework is among the most notable findings of this chapter, as suggested by experimental evidence. While lacking the sophistication of FPFH descriptors, the mere height coordinates of a point cloud lead to a more discriminative surface signature in natural environments. FPFH descriptors are better suited for urban environments.

Chapter 6 Urban road terrain classification using compositional high order pattern potentials (CHOPP)

6.1 Introduction

Road classification in a typical urban scenario benefits from the integration of prior knowledge into model predictions. Existing works make use of priors to predict either scalar labels or structured under the umbrella of semantic segmentation of atomic units. Among the works with scalar labelling a road location prior is most frequently used. For example it is assumed that the road is located in the bottom part of the image (Alvarez *et al.* 2012a) or below the horizon line and towards the vanishing point (Alvarez, Gevers and Lopez 2010). As an alternative to such hard assumptions, road location prior has been learned in the work of Alvarez *et al.* (2013) by using ground truth annotations of training examples. Structured prediction allows the integration of prior knowledge, towards the labelling of road atomic units, in different ways. One prevalent approach in the literature is to assume smooth labelling and attempt to find ways to preserve discontinuities (Alvarez *et al.* 2012b), (Passani, Yebes and Bergasa 2014), (Sturgess *et al.* 2009), (Zhang, Wang and Yang 2010). Loosely these priors reflect the knowledge that atomic units with similar saliency are likely just parts of the same object (e.g. road). Tarmac as a terrain class falls under the enlarged family of possible road surfaces that a

car might adhere to at any given time and it can hugely benefit not just from smoothness but also shape information since tarmac roads are designed for vehicles with Ackermann steering. To this end two additional models have been derived from the CRF models of Chapter 5, resulting in substantial improvements over the CRF baseline and other existing work benchmarked on the KITTI dataset. These additional methods impose a global shape prior for tarmac roads on top of the existing smoothness prior. The CRF can easily distinguish between {road} and {other} classes (possibly traffic participants such as cars) because their superpixels have quite different appearance and surface signatures. However it is much harder to achieve correct labelling when different classes emit similar appearance and surface signatures. In such case a global shape prior improves the semantic segmentation quality on the basis that roads exhibit shape in a typical urban environment.

6.2 Related work

Introduced in the context of semantic segmentation of face images, the global and local (GLOC) model auguments a sparsely connected pairwise CRF with Boltzmann machine shape priors (Kae *et al.* 2013). Work published simultaneously by Yujia, Tarlow and Zemel (2013) defines a class of compositional high order pattern potentials (CHOPP) that can be used to augment the conventional CRF. It has been identified that the RBM is a special case of CHOPP and therefore the GLOC model is equivalent to a CHOPP augmented CRF. There are however some differences between the works of Kae *et al.* (2013) and Yujia, Tarlow and Zemel (2013). For instance the atomic units are superpixels in the work of Kae *et al.* (2013) and pixels in the work of Yujia, Tarlow and Zemel (2013) which render the virtual pooling layer of GLOC unnecessary since the number of pixels remains constant in all images. In this case the labels of all the atomic units map directly onto the visible units of the RBM. The other notable difference is that Kae *et al.* (2013) modelled multinomial label shapes while Yujia, Tarlow and Zemel (2013)

modelled binary label shapes. The GLOC model will be utilised for semantic segmentation of road terrain scenes on KITTI road dataset albeit for binary classification of superpixel atomic units into road and non-road. Such model is able to strike a good balance between consistent labelling and road silhouette or global shape. Loosely the model formulation integrates the prior knowledge that units of similar saliency are likely just parts of the same object (i.e. should have the same label) and but labelling must also conform to a certain pattern of global shape.

6.3 CHOPP augmented CRF

Going beyond adjacent atomic units (i.e. using long range dependencies) allows the predictions of a structured model such as the CRF framework to benefit from more contextual information and two different approaches exist:

- Augmenting sparsely connected CRFs with higher order potential terms in addition to pairwise potentials such as smoothness in higher order neighbourhoods and more recently CHOPP.

- Fully connecting the graph in order to better capture relations between regions with different labels in addition to smoothness.

Consider the general class of compositional high order pattern potentials that is formulated by Yujia, Tarlow and Zemel (2013) and used to model binary label shapes. An equivalence between such potentials and the RBM is established:

$$f_T(Y) = -T \log\left(\sum_H exp\left(\frac{1}{T}\sum_{k=1}^{K}\left(b_k + \sum_{r=1}^{R} w_{rk} y_r\right) h_k\right)\right) \tag{6.1}$$

Here K represents the number of hidden variables and R represents the number of visible variables while the summation over H is a marginalisation over all possible configurations of hid-

den variables. By setting the temperature parameter $T = 1$ this becomes equivalent to the RBM high order potential.

$$f_{T=1}(Y) = -\log\left(\sum_{H} exp\left(\sum_{k=1}^{K}\left(b_k + \sum_{r=1}^{R} w_{rk} y_r\right) h_k\right)\right) \tag{6.2}$$

$$f_{T=1}(Y) = -\sum_{k=1}^{K} \log\left(1 + exp\left(b_k + \sum_{r=1}^{R} w_{rk} y_r\right)\right) \tag{6.3}$$

Adding this CHOPP alongside a bias term directly to the energy function of a sparsely connected pairwise CRF (i.e. with unary potentials ψ^u and pairwise potentials ψ^p) yields a conditional distribution of labells Y given features X which is equivalent to the GLOC model of Kae *et al.* (2013).

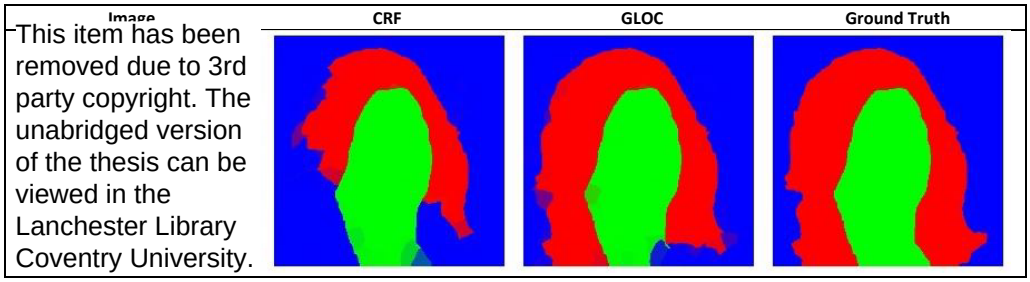
$$P(Y|X) = \frac{1}{Z(X)} exp\left(\psi^u + \psi^p + \sum_{r} c_r y_r + f_{T=1}(Y)\right) \tag{6.4}$$

$$P(Y|X) = \sum_{H} P(Y, H|X) \tag{6.5}$$

$$P(Y, H|X) = \frac{1}{Z(X)} exp\left(\psi^u + \psi^p + \sum_{r} c_r y_r + \sum_{rk} w_{rk} y_r h_k + \sum_{k} b_k h_k\right) \tag{6.6}$$

6.3.1 GLOC model

Table 6.1 Semantic segmentation results reproduced from Kae (2014). Colour coding shows green for skin, red for hair and blue for background. In this 3 class labelling problem CRF is shown to produce smooth labelling. GLOC enforces not only smoothness but also shape resulting in a more realistic labelling than CRF when compared to the ground truth. Superpixels act as 2D atomic units.



GLOC (i.e. global and local) has initially been proposed in Kae *et al.* (2013) and consists of a CRF augmented with CHOPP to model multinomial label shapes. In particular, it has been used in the context of facial recognition cast as semantic segmentation of skin, hair and background (Table 6.1). Its specifics will be discussed here whereby the formalisms of Chapter 3 will be carried forward and updated for clarity, to include details about learning and prediction as described by Kae (2014). To label a collection of segments, the most probable labels must be selected given the node and edge features as well as the learned parameters Γ, Ψ, W, B, C.

$$P_{GLOC}(Y|X) \propto \sum_{H} exp\big(-Energy_{GLOC}(Y,H,X)\big) \tag{6.7}$$

$$Energy_{GLOC}(Y,H,X) = Energy_{CRF}(Y,X) + Energy_{RBM}(Y,H) \tag{6.8}$$

$$Energy_{RBM}(Y,H) = -\sum_{r=1}^{R}\sum_{l=1}^{L} c_{rl}y_{rl} - \sum_{r=1}^{R}\sum_{l=1}^{L}\sum_{k=1}^{K} w_{rlk}y_{rl}h_k - \sum_{k=1}^{K} b_k h_k \tag{6.9}$$

In the energy formulation of the RBM $y_r \in \{0,1\}^L$ represent the visible units and $h_k \in \{0,1\}$ the hidden units while c_{rl} and b_k are their corresponding bias. The weights between visible and hidden units are $W \in \mathbb{R}^{R\times L\times K}$. This would suffice (Yujia, Tarlow and Zemel 2013) except for the fact that RBM needs a fixed number of visible nodes R and the number of superpixels $S^{(I)}$ might be different across different I. To deal with such problem Kae *et al.* (2013) introduced a virtual pooling layer that maps each superpixel label node into a fixed layer of visible nodes (on a grid with R elements) deterministically, using a projection matrix of size $R \times S$ containing elements p_{rs}.

$$p_{rs} = \frac{Area\big(Region(s) \cap Region(r)\big)}{Area\big(Region(r)\big)} \tag{6.10}$$

$$\sum_{s=1}^{S} p_{rs} = 1 \tag{6.11}$$

$$\bar{y}_{rl} = \sum_{s=1}^{S} p_{rs} y_{sl} \tag{6.12}$$

$$Energy_{RBM}(Y,H) = -\sum_{r=1}^{R}\sum_{l=1}^{L} c_{rl}\bar{y}_{rl} - \sum_{r=1}^{R}\sum_{l=1}^{L}\sum_{k=1}^{K} w_{rlk}\bar{y}_{rl}h_k - \sum_{k=1}^{K} b_k h_k \tag{6.13}$$

Parameter learning given some training data $\{Y^{(m)}, X^{(m)}\}_{m=1}^{M}$ consisting of M segmented images is performed by maximising the conditional log likelihood however Kae *et al.* (2013) advocate in favour of pretraining the individual components first corresponding to LR, CRF and RBM (training of the latter requires contrastive divergence (Hinton 2002) to approximate the parameter gradients). While the training and inference procedures of the other components have been outlined in Chapter 3, those associated with RBMs will be introduced here for the sake of completeness. If parameters are learned, inference in RBMs is done by performing a block Gibbs sampling. This requires sampling of all hidden and then all visible units, given the other units. In general, learning of the RBM parameters involves maximising the log likelihood of some M training data $\{Y^{(m)}\}_{m=1}^{M}$.

$$L = \max_{W,B,C} \sum_{m=1}^{M} \log\left(\sum_{H} P_{RBM}(Y^{(m)}, H)\right) \tag{6.14}$$

$$\frac{\partial L}{\partial w_{rlk}} = \left(\frac{1}{M}\sum_{m=1}^{M} y_{rl}^{(m)} P\left(h_k | y^{(m)}\right)\right) - P(y_{rl}, h_k) \tag{6.15}$$

$$\frac{\partial L}{\partial b_k} = \left(\frac{1}{M}\sum_{m=1}^{M} P\left(h_k | y^{(m)}\right)\right) - P(h_k) \tag{6.16}$$

$$\frac{\partial L}{\partial c_{rl}} = \left(\frac{1}{M}\sum_{m=1}^{M} y_{rl}^{(m)}\right) - P(y_{rl}) \tag{6.17}$$

Once the individual components are pretrained, the CHOPP augmented CRF (i.e. GLOC) is trained to maximize the conditional log likelihood.

$$L = \max_{\Gamma,\Psi,W,B,C} \sum_{m=1}^{M} \log P_{GLOC}(Y^{(m)}|X^{(m)}) \tag{6.18}$$

As pointed by Kae *et al.* (2013), the variational parameters μ_{sl} and γ_k corresponding to posterior labelling estimates of superpixels and estimates of the hidden units can be updated over a number of iterations indexed by i (until convergence or predefined) using mean field inference. Node, edge and RBM energies are as follows:

$$f_{sl}^{node}(X_V, \Gamma) = \sum_{d} x_{sd}\Gamma_{dl} \tag{6.19}$$

$$f_{sl}^{edge}(\mu; X_E, E, \Psi) = \sum_{j:(s,j)\in E} \sum_{l',e} \mu_{jl'}\,\Psi_{ll'e}x_{sje} \tag{6.20}$$

$$f_{sl}^{RBM}(\gamma; \{p_{rs}\}, W, C) = \sum_{r,k} p_{rs}(w_{rlk}\gamma_k + c_{rl}) \tag{6.21}$$

Subsequent updates of the variational parameters (Kae *et al.* 2013) are preceded by initialisation as follows:

$$\mu_{sl}^{(0)} = \frac{exp\left(f_{sl}^{node}\right)}{\sum_{l'} exp\left(f_{sl'}^{node}\right)} \tag{6.22}$$

$$\gamma_k^{(0)} = sigmoid\left(\sum_{r,l}\left(\sum_{s} p_{rs}\,\mu_{sl}^{(0)}\right)w_{rlk} + b_k\right) \tag{6.23}$$

$$\mu_{sl}^{(i+1)} = \frac{exp\left(f_{sl}^{node} + f_{sl}^{edge}\left(\mu^{(i)}\right) + f_{sl}^{RBM}\left(\gamma_k^{(i)}\right)\right)}{\sum_{l'} exp\left(f_{sl'}^{node} + f_{sl'}^{edge}(\mu^{(i)}) + f_{sl'}^{RBM}\left(\gamma_k^{(i)}\right)\right)} \tag{6.24}$$

$$\gamma_k^{(i+1)} = sigmoid\left(\sum_{r,l}\left(\sum_{s} p_{rs}\mu_{sl}^{(i+1)}\right)w_{rlk} + b_k\right) \tag{6.25}$$

6.3.2 Experiments

Experiments have been run and benchmarked on the challenging KITTI road dataset across all categories namely {urban marked, urban multiple marked lanes, urban unmarked} road using

all 289 training and 290 test images just as in Chapter 5. The atomic units are SLIC generated superpixels and the models CRF + [daisy 2, height] and CRF + [daisy 2, FPFH] are augmented with global shape priors (GSP) resulting in substantial relative improvements (i.e. over the CRF baselines before augmentation with CHOPP) as well as improvements over various other methods that are publicly ranked on the KITTI dataset. The acronyms under which the performance of these models can be publicly inspected are SCRFHGSP and SCRFFPFHGSP. Again these initials stand for the key elements that the model is composed of (e.g. Superpixels/Supervoxels under a Conditional Random Field with saliency from Fast Point Feature Histograms and Global Shape Priors). The GLOC implementation of Kae *et al.* (2013) has been used throughout this experiment. The reader is invited to inspect Appendix B, code snippet B.7 showing the general steps and routines utilised for the experiments of this chapter. Just as in the case of LR and CRF, this code snippet requires various components present in the GLOC source code in order to work properly.

6.3.2.1 Features

Features are similar to those used in Chapter 5 for the CRF + [daisy 2, height] and CRF + [daisy 2, FPFH] models without the position feature and correspond to nodes (i.e. superpixels) and edges (i.e. adjacent superpixels) of a random field. Node features capture the saliency of colour, texture and stereo surfaceness at each superpixel while the edge features are metrics of similarity between the saliency of adjacent superpixels. The common approach towards extracting node features is bag of words, a vector quantization technique that models saliency statistics across prototype vectors. While in general more prototypes enable better saliency representation there is also the risk of negatively impacting performance by overrepresentation (i.e. too many prototypes). In this work the number of prototypes has been fielded after heuristically experimenting with values around common quantization levels reported in the literature.

6.3.2.1.1 Node features

- Colour histogram

  Each superpixel is represented by a normalised colour histogram using bag of colour words. This approach is essentially counting pixel occurrences across 64 bins corresponding to nearest Lab colour prototypes. To obtain the colour prototypes a number of training images are set aside, converted to Lab and concatenated. This large collage of images becomes the search space of a K-means algorithm with 64 seed points initialised probabilistically (Arthur and Vassilvitskii 2007). After convergence, the centroids of those 64 clusters become the colour prototypes.

- Texture histogram

  Texture saliency of every superpixel is captured using bag of texture words obtained from daisy descriptors (previously named daisy 2 simply to denote the histogram variant with more daisy prototypes i.e. 300 bins). Again daisy descriptors are collaged together after being extracted at every pixel from a number of training images. Subsequently the same variant of K-means with probabilistic seeding is able to output 300 daisy prototypes in the form of cluster centroids. At each superpixel texture saliency is then represented as a descriptor count across bins corresponding to nearest descriptor prototypes.

- 3D information

  What sets the two models apart namely the SCRFHGSP and SCRFFPFHGSP is how the surface saliency is encoded as part of the node features for each superpixel using stereo vision. SCRFHGSP uses bag-of-height-words. A number of training point clouds are set aside and all reconstructed points have their height components combined into a new array. This is to create a pool of possible height values within the training set. Clustering these height values with K-means learns 64 height prototypes towards vec-

tor quantization of surface saliency at every superpixel. Similarly, SCRFFPFHGSP uses bag-of-FPFH-words extracted from training point clouds towards vector quantization with 300 prototypes.

6.3.2.1.2 Edge features

- Colour

  The mean Lab colour is computed for each SLIC superpixel of an image and the l_2 distance is taken between the mean Lab colours of neighbouring superpixels. The smaller this distance the more similar their colour saliency and the more likely the superpixels are to be part of the same object (e.g. road).

- Texture

  Texture saliency similarity between neighbouring superpixels is computed as the chi-squared distance between their texture histograms h_1 and h_2 similar to Huang, Narayana and Learned-Miller (2008), Kae *et al.* (2013). Texture histograms have 300 dimensions since daisy 2 is used to describe each superpixel. Again, the more similar their texture saliency, the more likely the superpixels are to be part of the same object class.

- 3D information

  Saliency similarity of 3D surfaceness between neighbouring superpixels is given by the intersection kernel (Barla, Odone and Verri 2003) which has been experimentally confirmed as the appropriate distance measure among a handful of candidates.

  In fact Rusu, Blodow and Beetz (2009) used the intersection kernel to show that FPFH alone are features able to discriminate among different primitive geometric surfaces of point clouds such as plane, sphere, cylinder, edge and corner. Depending on how the 3D information obtained via stereo vision is encoded as superpixel or node feature

(i.e. using height or FPFH descriptors), this edge feature will correspond to either the SCRFHGSP or SCRFFPFHGSP model.

6.3.2.2 Evaluation

The KITTI road evaluation benchmark makes use of specific pixel based metrics in BEV space namely, maximum F_1-measure, average precision, precision, recall, false positive rate and false negative rate. In particular the F_1 based metric is used towards ranking the methods. The reader is referred to the evaluation section of Chapter 5 where the mathematical formulations of these measures are provided.

6.3.2.3 Comparison to prior work

The proposed methods attain large improvements over their CRF baseline predecessors (i.e. SCRFH and SCRFFPFH can be inspected alongside their CHOPP augmented counterparts SCRFHGSP and SCRFFPFHGSP at http://www.cvlibs.net/datasets/kitti/eval_road.php). In addition to this comparison, the benchmark tables highlight relative improvements with respect to other recent methods in all sections namely {urban marked, urban multiple marked lanes, urban unmarked} road. While various other authors have submitted results of their methods anonymously, tables reproduced here include the relative improvements only over those works backed by available publications. More specifically, Table 6.2 presents the accuracy scores attained by the aforementioned methods on the urban marked road testing subset of the KITTI road evaluation dataset, alongside the scores of other benchmark participants. Similarly, Table 6.3 and Table 6.4 show the accuracy scores attained across the urban multiple marked and urban unmarked testing subsets respectively. To evaluate (i.e. benchmark) a method, one needs to upload the classification results or confidence maps produced by such method across a KITTI test dataset. The KITTI road evaluation server will then process these results and convert them into accuracy scores internally. The ground truth labelling of the test

dataset in known only to the server. Subsequently, the method is ranked and displayed online alongside other such endeavours. Improvements with respect to the CRF baselines in all benchmark categories can also be visually inspected. Table 6.5 shows visual results across the urban marked road test subset labelled by CRF and CHOPP augmented CRF, both algorithms using bag of height words as surface saliency. Table 6.6 shows visual results across the same test subset and algorithms but this time using bag of FPFH words as surface saliency. In a similar fashion, Table 6.7 and Table 6.8 are dedicated to the urban multiple marked lanes test subset while Table 6.9 and Table 6.10 are dedicated to the urban unmarked test subset. Visual results across the same test subset (i.e. {urban marked, urban multiple marked lanes or urban unmarked}) appear similar regardless of the surface saliency of choice. Large improvements are seen due to the CHOPP augmented CRF algorithm being able to enforce road silhouette. In this context the benefits of incorporating prior knowledge into model predictions in urban environments are apparent. Up to date full tables and ranked methods across all road categories are accessible online. Note that since the ground truth of the KITTI test set is not publicly available, the evaluation methodology previously used in Chapter 4 and Chapter 5 cannot be extended here.

Table 6.2 Urban marked road. MaxF: Maximum F1-measure, AP: Average precision, PRE: Precision, REC: Recall, FPR: False Positive Rate, FNR: False Negative Rate

Method	Setting	MaxF	AP	PRE	REC	FPR	FNR
…	…	…	…	…	…	…	…
SCRFFPFHGSP	stereo	83.73%	72.89%	82.13%	85.39%	8.47%	14.61%
HistonBoost (Vitor, Victorino and Ferreira 2014)	stereo	83.68%	72.79%	82.01%	85.42%	8.54%	14.58%
SCRFHGSP	stereo	83.25%	72.41%	81.54%	85.04%	8.77%	14.96%
BL (Fritsch, Kuhnl and Geiger 2013)	monocular	82.24%	85.30%	79.44%	85.24%	10.05%	14.76%
BM (Wang, Fremont and Rodriguez 2014)	stereo	78.90%	66.06%	69.53%	91.19%	18.21%	8.81%
SPlane (Einecke and Eggert 2014)	stereo	78.19%	76.41%	72.03%	85.50%	15.13%	14.50%
CN24 (without spatial prior) (Brust *et al.* 2015)	monocular	76.28%	79.29%	72.44%	80.55%	13.96%	19.45%
CN (Alvarez *et al.* 2012a)	monocular	73.69%	76.68%	69.18%	78.83%	16.00%	21.17%
ARSL-AMI (Passani, Yebes and Bergasa 2014)	monocular	71.97%	61.04%	78.03%	66.79%	8.57%	33.21%
SCRFFPFH	stereo	70.78%	64.76%	83.88%	61.22%	5.36%	38.78%
SCRFH	stereo	69.34%	60.30%	84.47%	58.81%	4.93%	41.19%
ANN (Vitor *et al.* 2013)	stereo	62.83%	46.77%	50.21%	83.91%	37.91%	16.09%

Table 6.3 Urban multiple marked lanes road. MaxF: Maximum F1-measure, AP: Average precision, PRE: Precision, REC: Recall, FPR: False Positive Rate, FNR: False Negative Rate

Method	Setting	MaxF	AP	PRE	REC	FPR	FNR
…	…	…	…	…	…	…	…
SCRFFPFHGSP	stereo	87.96%	83.16%	90.01%	86.01%	10.50%	13.99%
SCRFHGSP	stereo	87.75%	83.53%	90.45%	85.21%	9.89%	14.79%
CN (Alvarez *et al.* 2012a)	monocular	86.21%	84.40%	82.85%	89.86%	20.45%	10.14%
SPlane (Einecke and Eggert 2014)	stereo	82.28%	82.83%	76.85%	88.53%	29.32%	11.47%
SCRFH	stereo	82.08%	80.37%	90.87%	74.83%	8.26%	25.17%
SPlane + BL (Einecke and Eggert 2014)	stereo	82.04%	85.56%	75.11%	90.39%	32.93%	9.61%
ANN (Vitor *et al.* 2013)	stereo	80.95%	68.36%	69.95%	96.05%	45.35%	3.95%
SCRFFPFH	stereo	79.75%	80.37%	90.87%	71.06%	7.85%	28.94%
BL (Fritsch, Kuhnl and Geiger 2013)	monocular	76.02%	78.82%	65.71%	90.17%	51.72%	9.83%

Table 6.4 Urban unmarked road. MaxF: Maximum F1-measure, AP: Average precision, PRE: Precision, REC: Recall, FPR: False Positive Rate, FNR: False Negative Rate

Method	Setting	MaxF	AP	PRE	REC	FPR	FNR
…	…	…	…	…	…	…	…
SCRFHGSP	stereo	81.21%	70.94%	81.24%	81.17%	6.11%	18.83%
SCRFFPFHGSP	stereo	80.78%	70.80%	81.07%	80.50%	6.13%	19.50%
BM (Wang, Fremont and Rodriguez 2014)	stereo	78.43%	62.46%	70.87%	87.80%	11.76%	12.20%
HistonBoost (Vitor, Victorino and Ferreira 2014)	stereo	74.19%	63.01%	77.43%	71.22%	6.77%	28.78%
SPlane + BL (Einecke and Eggert 2014)	stereo	74.02%	79.61%	65.15%	85.68%	14.93%	14.32%
SPlane (Einecke and Eggert 2014)	stereo	73.30%	69.11%	65.39%	83.38%	14.38%	16.62%
CN (Alvarez *et al.* 2012a)	monocular	72.25%	66.61%	71.96%	72.54%	9.21%	27.46%
ARSL-AMI (Passani, Yebes and Bergasa 2014)	monocular	70.33%	61.97%	83.33%	60.84%	3.97%	39.16%
BL (Fritsch, Kuhnl and Geiger 2013)	monocular	69.50%	73.87%	65.87%	73.56%	12.42%	26.44%
SCRFFPFH	stereo	64.97%	55.97%	82.13%	53.74%	3.81%	46.26%
SCRFH	stereo	64.00%	55.99%	82.16%	52.41%	3.71%	47.59%
ANN (Vitor *et al.* 2013)	stereo	54.07%	36.61%	39.28%	86.69%	43.67%	13.31%

Table 6.5 Urban marked road labelled by CRF (left) and CHOPP augmented CRF (right). Surface saliency within the point cloud is captured using bag of height words

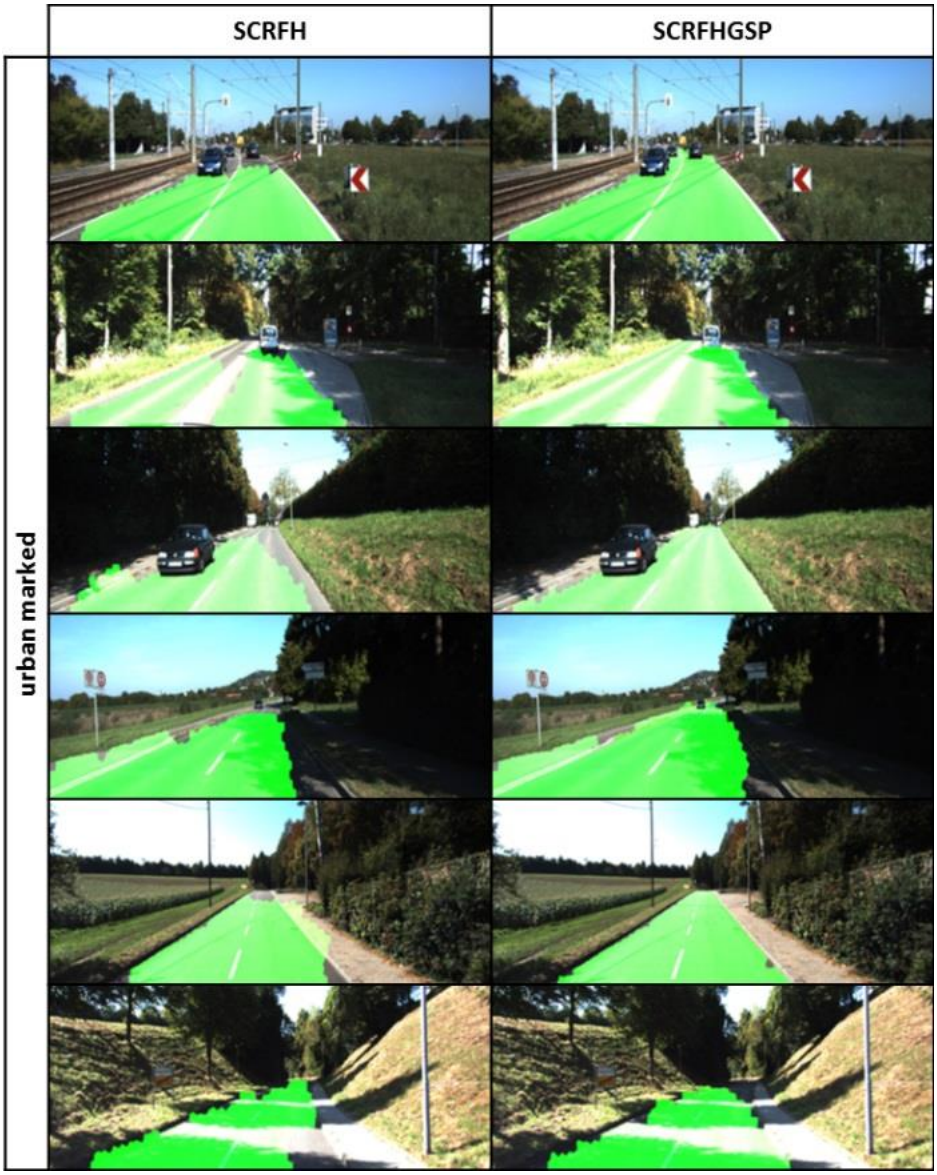


Table 6.6 Urban marked road labelled by CRF (left) and CHOPP augmented CRF (right). Surface saliency within the point cloud is captured using bag of FPFH words

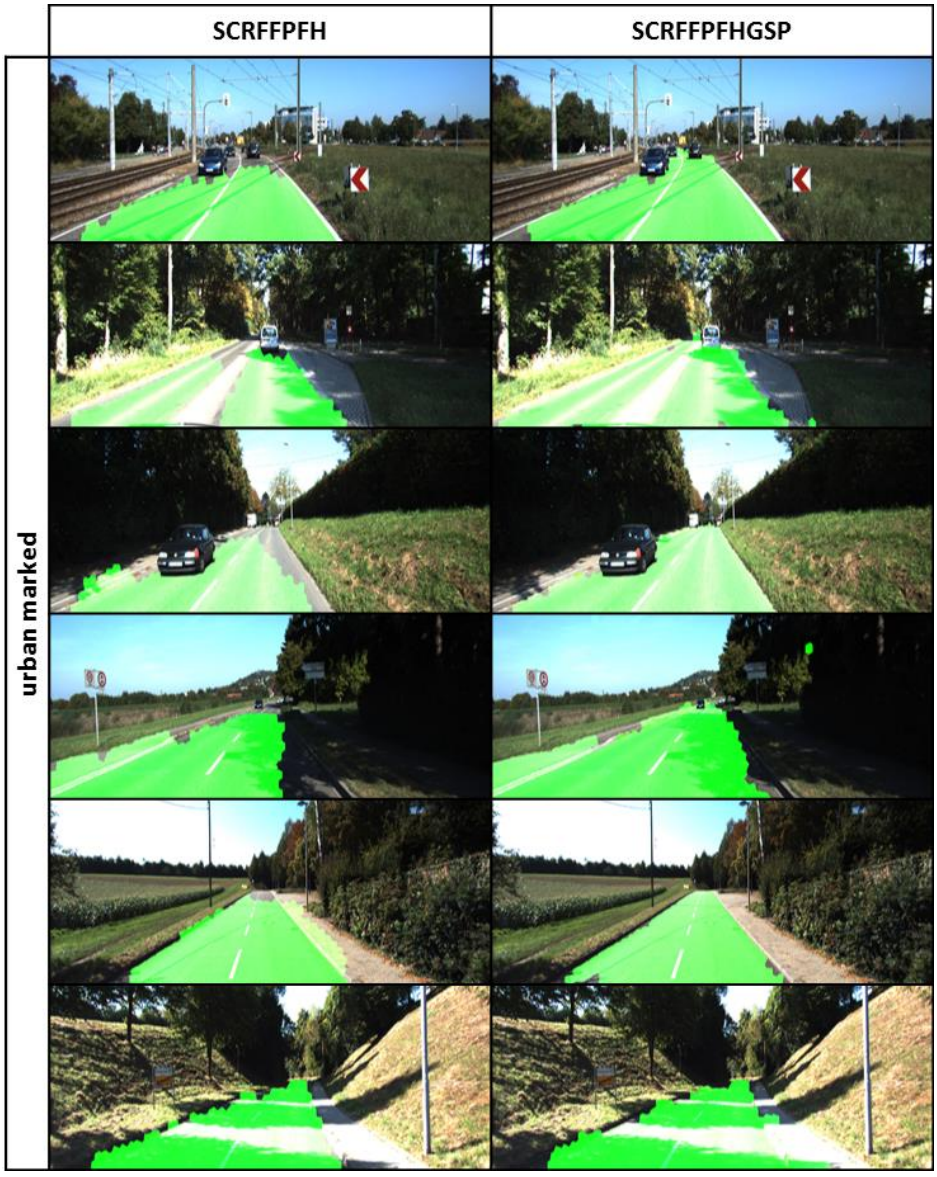


Table 6.7 Urban multiple marked lanes road labelled by CRF (left) and CHOPP augmented CRF (right). Surface saliency within the point cloud is captured using bag of height words

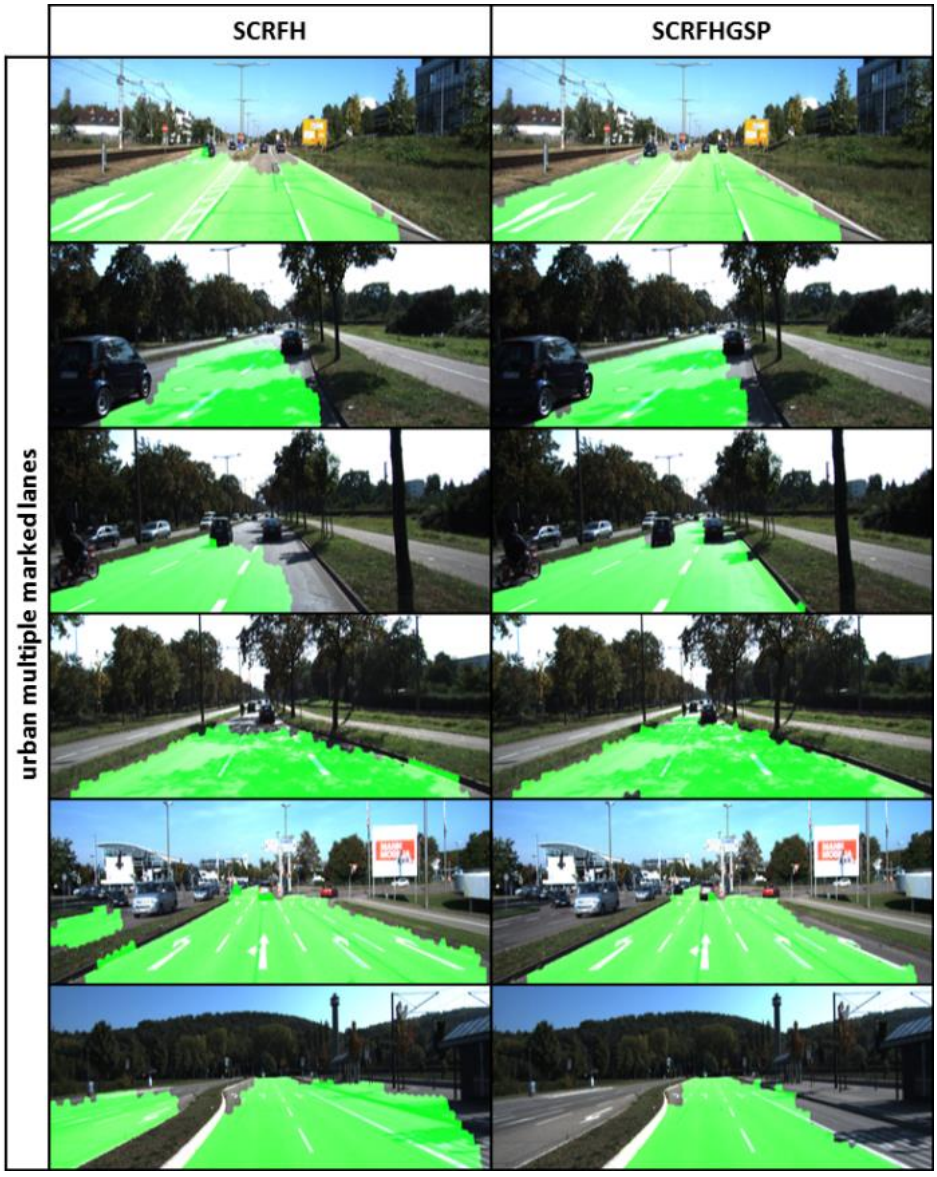


Table 6.8 Urban multiple marked lanes road labelled by CRF (left) and CHOPP augmented CRF (right). Surface saliency within the point cloud is captured using bag of FPFH words

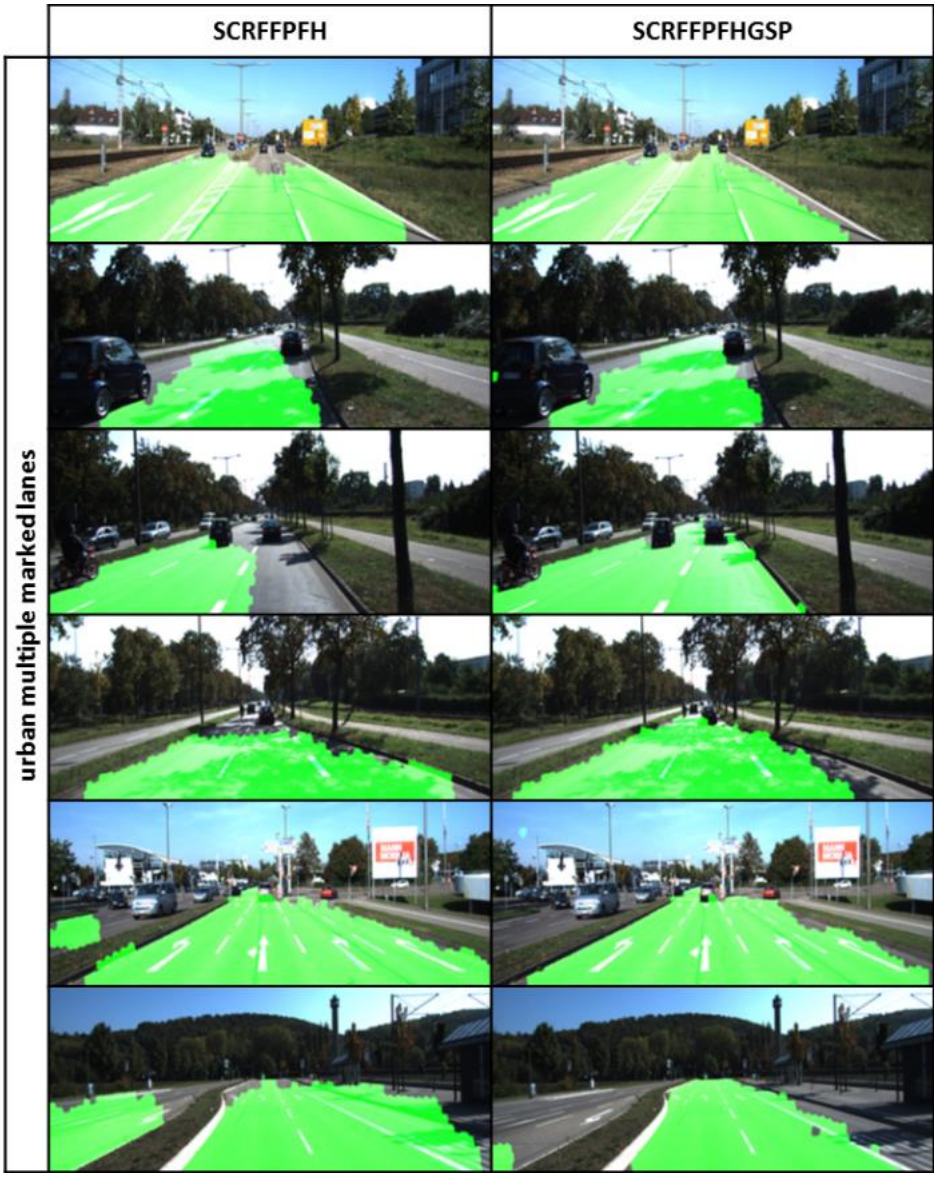


Table 6.9 Urban unmarked road labelled by CRF (left) and CHOPP augmented CRF (right). Surface saliency within the point cloud is captured using bag of height words

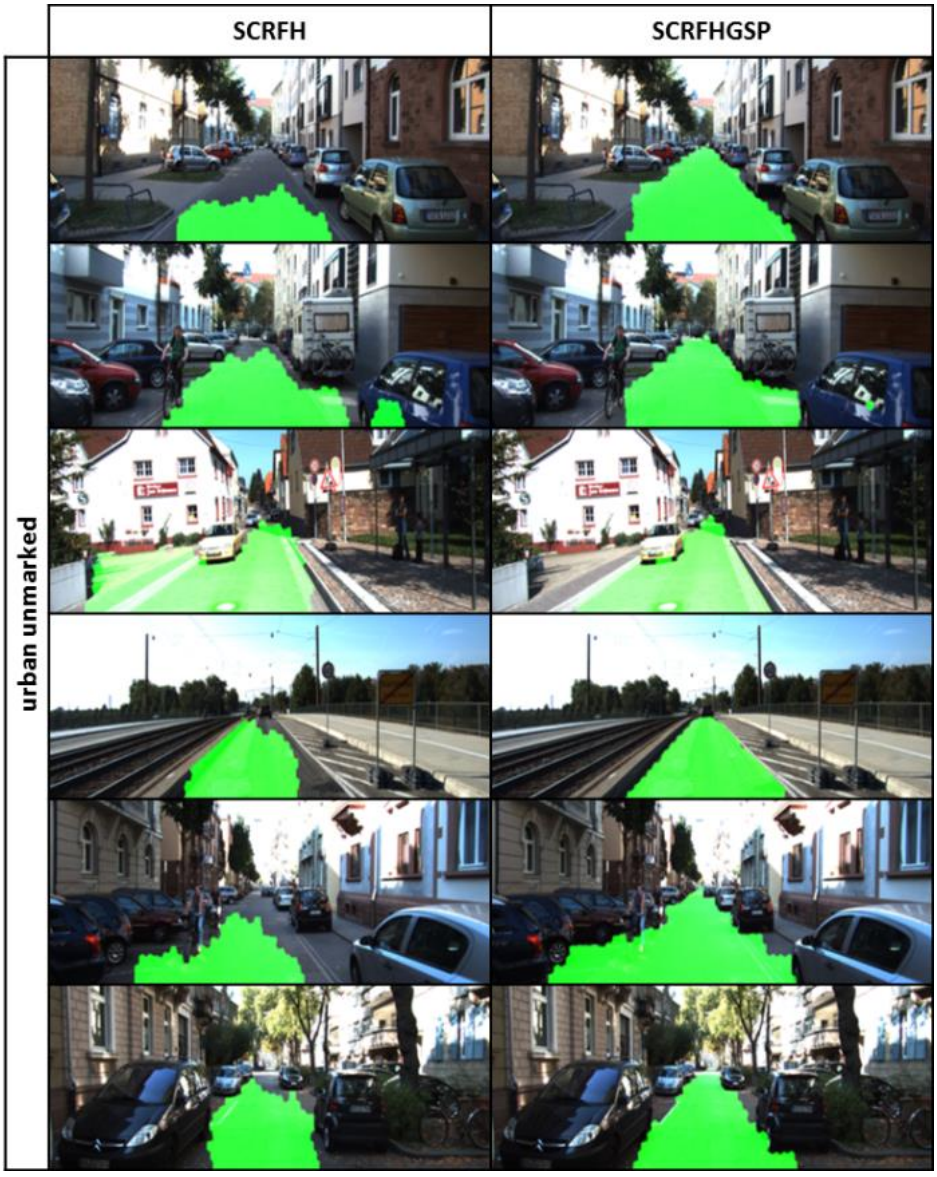
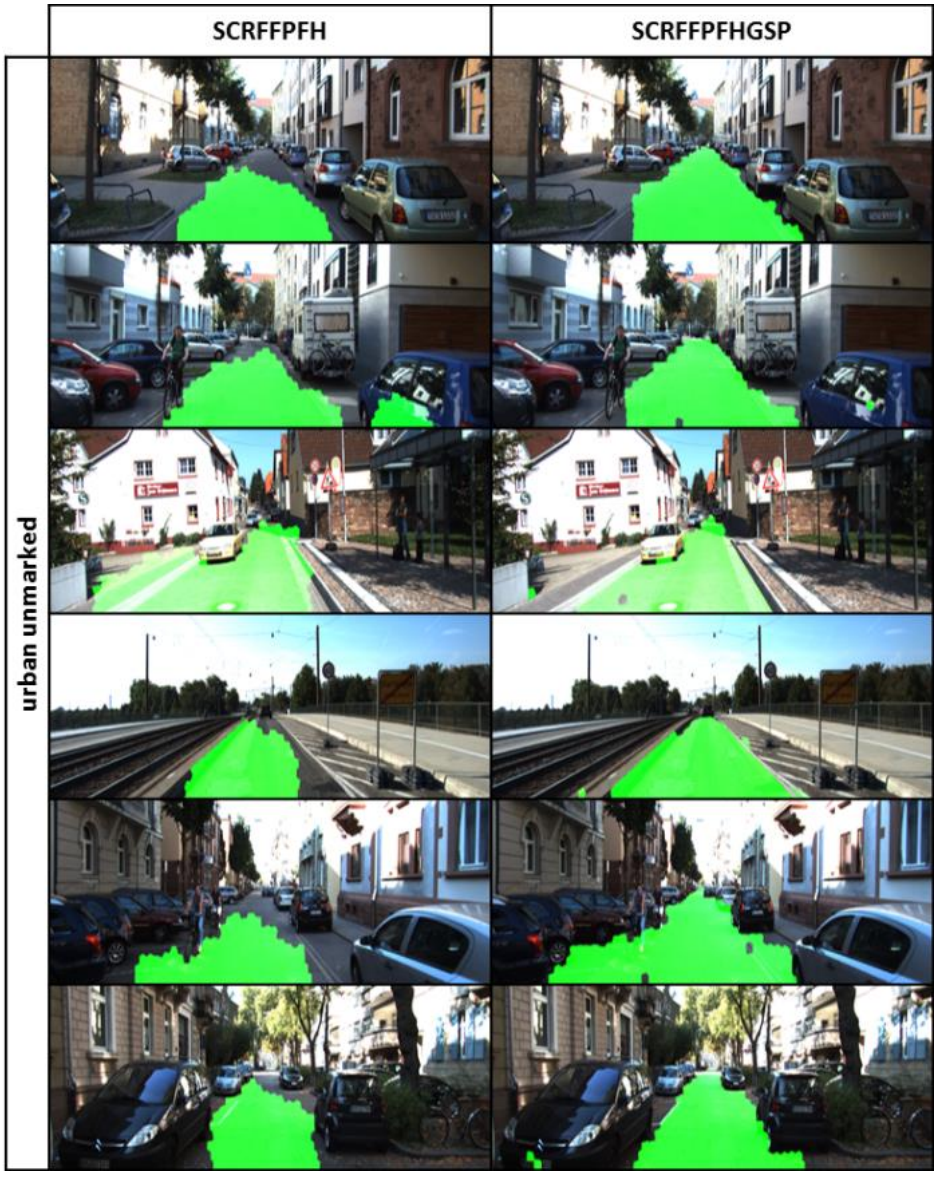


Table 6.10 Urban unmarked road labelled by CRF (left) and CHOPP augmented CRF (right). Surface saliency within the point cloud is captured using bag of FPFH words



6.4 Discussion

CHOPP augmented CRFs are still in their infancy but show great promise for a variety of semantic segmentation tasks (Gould and He 2014). Here, the recently proposed GLOC model (Kae *et al.* 2013) has been used for binary labelling (i.e. road or non-road) of superpixels depicting urban road scenes. Unlike natural environments, urban road scenes exhibit structure and many objects of interest for ADAS applications exhibit shape (e.g. cars, pedestrians). Perhaps not as intuitive but urban roads themselves exhibit shape too, even though tarmac falls into the *stuff* category and as a superclass (i.e. material) is typically amorphous. This is based on a simple real world remark regardless of the sensing technology of choice, namely that urban tarmac roads are designed for vehicles with Ackermann steering. Yujia, Tarlow and Zemel (2013) argue that performance of CHOPP augmented CRFs drops on high variability data. However, large performance gains with respect to the CRF baseline are attained here enabling accurate road recognition. Using the GLOC model for structured labelling of superpixel atomic units combines two types of prior knowledge (i.e. smoothness and global shape) into a single objective. When used jointly these priors ensure that road labelling is both locally consistent as well as globally coherent, resembling a likely road silhouette. As far as capturing surface saliency from point clouds and attributing it to each superpixel individually both height and FPFH exhibit similar discriminative power, with bag of FPFH words performing marginally better than height across the overall test set. At the time of evaluation the overall ranks of SCRFFPFHGSP and SCRFHGSP were 13 and 14 respectively (out of 31 submitted methods) on the challenging KITTI road dataset. This chapter has presented the most notable contribution of the thesis. The novelty consists of using CHOPP augmented CRF towards binary labelling of urban environments into road and non-road semantics. In fact, this novelty is best observed by inspecting the literature review chapter, Table 2.1 and Table 2.2.

Chapter 7 Conclusion and future work

7.1 Achieved results

In pursuit of a terrain response ADAS, prospective terrain recognition has been tackled using the mid-level vision task of semantic segmentation. This is in line with the main research objective of this thesis. Furthermore, desirable semantics considered include {grass, trees, sky, dirt, gravel, shrubs, tarmac and void}. Presegmenting images into small size superpixels results in a collection of atomic units that can adhere well to boundaries and overcome the rigidity of grid approaches while reducing the number of hypothesis evaluations or the complexity of a graphical model. In search of a suitable terrain recognition approach, several predictors (i.e. SVM, RF, LR and CRF) and terrain saliency flavours have been tested. This is in line with the research objective seeking a suitable machine learning scheme and feature representation. Selections of appropriate texture descriptors as well as their granularity lead to a superior semantic segmentation based on the CRF framework. Due to the unstructured character of a natural environment only a pairwise smoothness prior is enforced on the final labelling in a sparsely connected random field. Such pairwise potentials are able to encapsulate the smoothness constraint on neighbouring atomic units (i.e. superpixels or supervoxels) by using different measures of dissimilarity based on colour, texture and features from stereo reconstruction. Naturally, with the introduction of stereo vision smoothness is not a suitable assumption for all *stuff* classes present in forward driving scenes. Semantic segmentation of a collection of trees for example is marginally worsened with structured output learning since their atomic units

can exhibit different surface signatures locally (e.g. tree trunks, branches and foliage). In this case smoothness tends to hold up for the other saliency components based on colour and texture appearance. With structured output learning, stereo vision is also able to reduce the reliance on some discontinuity preserving measures based purely on appearance such as the probability of boundary. In fact the presence of strong shadows amounts to false boundaries that are best discarded. This approach is generic in the sense that it can be used to label terrain types in both natural and urban environments. However, unlike the natural environment, its urban counterpart is heavily constrained and structured. Often times the road terrain has shape since street roads are designed for vehicles with Ackermann steering. Exploiting such global shape information leads to superior semantic segmentation as demonstrated in the KITTI road benchmark. To this end previous pairwise CRF formulations are augmented with compositional high order pattern potentials (CHOPP) (Yujia, Tarlow and Zemel 2013) thus achieving two objectives with a single framework: global shape and local coherence (GLOC) as proposed by Kae *et al.* (2013). It is important that various road shapes and traffic situations are accounted for, as predictions across unseen samples (i.e. new images) will be smooth and resemble the most likely road shapes (Figure 7.1). The stated research objective of incorporating prior knowledge about the problem domain into model predictions has been attained by using the inherent smooth prediction of a CRF classifier to label natural environments and by using a CHOPP augmented CRF classifier to label urban environments (albeit modelling binary label shapes). The latter is able to enforce both smoothness and shape. The other stated research objectives demanding suitable feature representations (in 2D and 3D realms) or visual and spatial terrain saliency have been attained as part of a bag of features approach. While good texture and surface representations have been attained using daisy descriptors and FPFH descriptors respectively, the pursuit of better descriptors or learning descriptors altogether remains an open objective.

Figure 7.1 Labelling of an unseen sample image. While appearance and surface saliency of most atomic units inside the red circle correspond to road/tarmac, labelling resembles a likely overall scene layout similar to those present in the training set

- Remark

From an ADAS application point of view, at the core of the proposed framework lies the CRF which assumes that road scene is drawn from a multivariate probability distribution and atomic units are piecewise smooth. Under this framework, other than descriptor prototypes, node $\Gamma \in \mathbb{R}^{L \times D_n}$ and edge $\Psi \in \mathbb{R}^{L \times L \times D_e}$ weights are all that is needed to make predictions when new input data is available. However, a particular dataset such as the JLR dataset (representative of the West Midlands area) does not fully span the variance of individuals within a class (e.g. grass, trees) since they are quite different from one geographical region to another. Fortunately, vehicles are preconfigured and tailored to match different demands from around the world. Moreover, the current trend suggests that future vehicles might ship with internet connectivity as standard. Hence model parameterisation can be received wirelessly as an update given the geographical area, much like the present smartphone updates. To further automate this process the geographical area can be determined using the internet connectivity and further refined using a GPS sensor thus reducing or removing intrusiveness of such ADAS completely.

7.2 Recommendations

Concerning the possibility of having a prospective terrain recognition ADAS on a vehicle equipped with vision (i.e. monocular and stereo), several recommendations can be made based on evidence emerging from the experiments of this thesis:

- Prospective terrain recognition is best achieved using the mid-level vision task of semantic segmentation, as opposed to object detection. Firstly, this allows for granular classification results that are more localised than a mere bounding region. Secondly, the semantic segmentation makes it easier for contextual information to be included.

- Consider features and algorithms when it comes to semantic segmentation of terrain classes. This is analogous to answering both questions: what and how to learn. Both routes can lead to superior accuracy.

- Terrain classes are of amorphous special extent and contain granularly repetitive patterns. Therefore, one needs to seek good visual saliency representation in the form of colour and most importantly texture.

- Consider representing texture using bag of visual words approach. Select a state of the art key-point descriptor (e.g. SIFT, daisy etc.) and perform vector quantisation across a number of prototypes to create a pattern for each segment. Relevant literature should be reviewed do determine the initial number of prototypes as this may vary from one key-point descriptor to another for a typical segment. At this point, it is worth experimenting with various quantisation levels in a close vicinity. There are significant accuracy gains to be made this way. Such vicinity can be regarded as a band outside of which texture becomes less discriminative.

- Refrain from using prior knowledge or assumptions when semantically segmenting the natural environment, with the exception of smoothness as part of a CRF algorithm. Terrain classes are smooth in terms of visual saliency and surface signature.

- Do consider using prior knowledge in addition to smoothness in order to semantically segment urban environment roads. Assuming that urban roads exhibit shape thus being fairly constrained is a sensible thing to do. Even more sensible would be to consider road shape variability in conjunction with a CHOPP augmented CRF.

- Select appropriate dissimilarity measures or edge features between segments (i.e. superpixels or supervoxels). There are various ways to compute the notion of distance between histograms (e.g. chi-square distance, Euclidean distance, intersection etc.) and likewise this can potentially impact accuracy. The reliance on visual dissimilarity measures decreases with the introduction of spatial dissimilarity measures, particularly if the visual ones are vulnerable to strong shadows.

- For semantic segmentation in 3D space use a supervoxelisation algorithm to partition the point cloud into segments if the ranging sensor produces dense and accurate point clouds. In the case of stereo reconstruction an equivalence between superpixels and supervoxels may be assumed given the superpixel reconstruction.

- Use bag of spatial words to summarise the surface signature corresponding to a supervoxel. Experimental evidence suggests that vector quantisation across FPFH prototypes is better suited for urban environments. Intuitively, tarmac roads are flat and the FPFH point descriptors have the ability to capture such surface saliency more accurately than simple height. Similarly, height may be utilised in a similar fashion to summarise surface saliency of segments in natural environments.

- The recommended classification algorithms should remain unchanged, namely a simple pairwise CRF for natural environments and a CHOPP augmented CRF for urban environments. The atomic units should however be regarded as supervoxels and node features should consist of a concatenation of histograms corresponding to visual saliency and surface saliency. Conversely, the edge features must now be concatenated with a surface dissimilarity measure. The intuition is that neighbouring supervoxels with similar visual and surface saliency (e.g. convexity, irregularity etc.) are likely part of the same class.

7.3 Future development

With regards to the methodology proposed to solve the sematic labelling problem several future research paths have been established given the current progress.

- Better stereo data can be obtained using state of the art stereo methods and cameras. Alternatively, LIDAR (or TOF) could be used to get more reliable depth information or the two sensors can be fused together (Badino, Huber and Kanade 2011), (Zhu *et al.* 2008). To give an example, after calibrating the sensors into a common reference, depth can be estimated as a weighted average of both sensors. Stereo cameras can provide depth information predominantly from low albedo regions (i.e. dark patches) and laser from everywhere else. In addition to that dense stereo estimation can compensate for laser's sparsity (Badino, Huber and Kanade 2011). The proposed framework would benefit from any improvement in the reliability of the real world estimation in two ways. Firstly, instead of estimating supervoxels solely in the image domain, the real world can help filter the initial segments based on some real world criteria (e.g. points with locally coherent normals, similar disparity range). Alternatively supervoxelisation can be performed exclusively in the real world coordinates (Douillard *et*

al. 2011b) though metrics describing point distance should be adaptive. Secondly, reliable features build on top of reliable sensor readings if they are to truly capture the underlying saliency of any class.

- To achieve day and night functionality for such an ADAS application, it is worth exploring multispectral cameras in the thermal IR as suggested by Manduchi *et al.* (2005).

- The proposed feature and algorithm configuration supports the inclusion of additional amorphous classes depending on their relevance in the road scene. This can be used to determine how a vehicle might negotiate other classes such as sand, snow and mud, to name a few. Conversely, some classes can be trimmed from the model all together (i.e. merge shrubs and trees).

- Instead of handcrafting features, a feature learning module could be used to learn good representation from raw data in both the image domain (such as a convolutional neural net) as well as in the real world domain discretised with point clouds (Lai, Bo and Fox 2014). For example, single layer networks have been used to learn domain specific features in an unsupervised way by Coates, Ng and Lee (2011).

- Temporal coherence between frames can be encouraged such as in the work of Kae (2014) by adding temporal potentials to the model. To this end the current frame predictions can be influenced by a number of past frames with contributions weighted in a decaying fashion. The application domain can be exploited even further to make this mechanism more adaptable. It can be observed that steering angle position and steering angle velocity (accessible via the CAN bus) have a direct influence on the amount of correlation between neighbouring frames, unless the environment is extremely ambiguous (e.g. flat desert or snow until vanishing line). The more steering action, the

less correlation and therefore a steeper decaying of weights can be applied at neighbouring frames.

- To increase the feature extraction speed all superpixels can be processed at the same time using multiple threads. Work presented here is highly parallelisable and amenable to an FPGA implementation (Johnston, Gribbon and Bailey 2004) thus opening the path for a real time application.

- The proposed framework can classify point clouds obtained using airborne ranging sensors such as LIDAR (Niemeyer, Rottensteiner and Soergel 2012) since point cloud descriptors are independent of the view point (i.e. FPFH). If such environments are unstructured (and piecewise smooth) a sparsely connected graph with pairwise smoothness prior would suffice. Alternatively if the environments are more structured, a higher order pattern potential such as one learned using RBM to model shape could improve classification accuracy.

Appendix A Models usage

Diagram A.1 From image (2D) to semantic segmentation via scalar predictions. As part of a preprocessing stage SLIC algorithm partitions the original image into superpixels. Feature extraction and scalar classification follow. The former relies on a concatenation of histograms capturing visual saliency (e.g. texture) of every superpixel while the latter relies on a scalar output learning algorithm of choice (i.e. SVM, RF or LR). To label an entire image all segments must be labelled individually. Each label is colour coded for better visualisation.

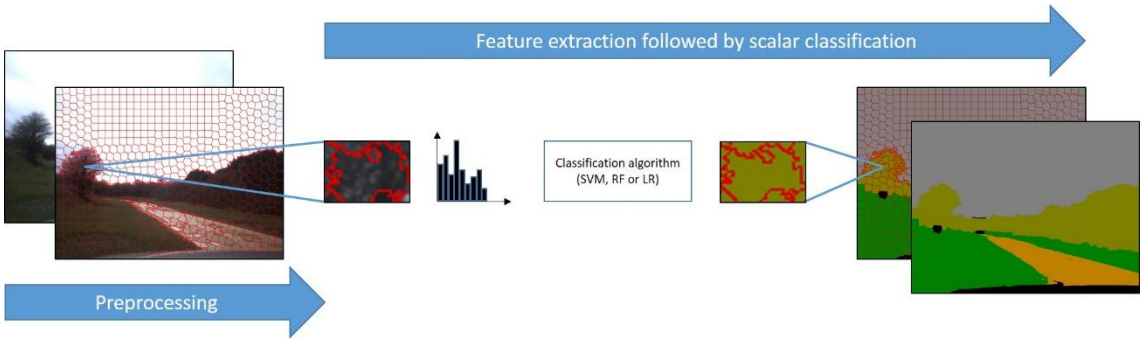


Diagram A.2 From image (2D) to semantic segmentation via structured prediction. As part of a preprocessing stage SLIC algorithm partitions the original image into superpixels. Feature extraction and structured classification follow. The former relies on two types of features: a concatenation of histograms capturing visual saliency (e.g. texture) of every superpixel (i.e. node features) and a concatenation of dissimilarity measures between adjacent superpixels (i.e. edge features or links between nodes). Classification is performed using a structured output learning algorithm of choice (i.e. CRF). All segments are classified by selecting the most likely labelling configuration across the entire image. Each label is colour coded for better visualisation.

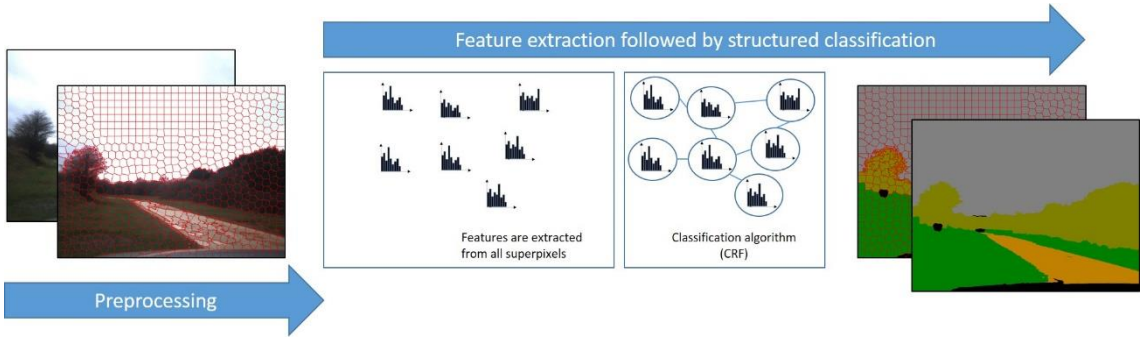


Diagram A.3 From image (2D) and point cloud (3D) to semantic segmentation via scalar predictions. The preprocessing stage is composed of two independent steps namely, superpixel partitioning (SLIC on the original image) and stereo reconstruction. Reconstructed SLIC superpixels span the subspace of supervoxels in 3D. Feature extraction and scalar classification follow. The former relies on a concatenation of histograms capturing visual saliency (e.g. texture) and surface saliency of every superpixel/supervoxel while the latter relies on a scalar output learning algorithm of choice (i.e. SVM, RF or LR). To label an entire image all segments must be labelled individually. Each label is colour coded for better visualisation.

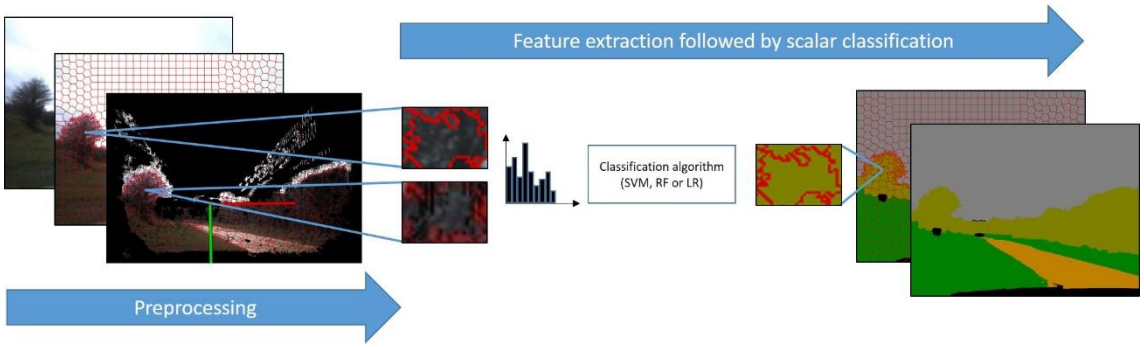
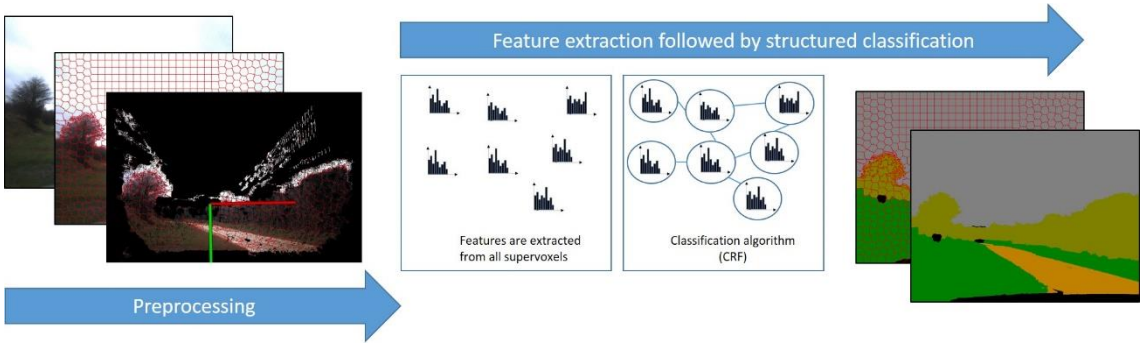


Diagram A.4 From image (2D) and point cloud (3D) to semantic segmentation via structured prediction. The preprocessing stage is composed of two independent steps namely, superpixel partitioning (SLIC on the original image) and stereo reconstruction. Reconstructed SLIC superpixels span the subspace of supervoxels in 3D. Feature extraction and structured classification follow. The former relies on two types of features: a concatenation of histograms capturing visual saliency (e.g. texture) as well as surface saliency of every superpixel/supervoxel (i.e. node features) and a concatenation of dissimilarity measures between adjacent superpixels/supervoxels (i.e. edge features or links between nodes). Classification is performed using a structured output learning algorithm of choice (i.e. CRF). All segments are classified by selecting the most likely labelling configuration across the entire image. Each label is colour coded for better visualisation.



Appendix B Code samples

Code snippet B.1 This Python code produces a single confusion matrix. It can be used in conjunction with any learning algorithm such as those presented in Chapter 3.

```
# predicted and true_labels are vectors known beforehand
# i.e. machine learning/algorithm predictions and ground truth respectively
# cm_location is the file location where confusion matrix is to be saved
# see http://stackoverflow.com/questions/5821125
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
Y_pred_prep= predicted
Y_test_prep= true_labels
labels= ["grass", "tree", "sky", "dirt", "gravel", "shrubs", "tarmac", "void"]
cm= confusion_matrix(Y_test_prep, Y_pred_prep)
conf_arr = cm
norm_conf = []
for i in conf_arr:
 a = 0
 tmp_arr = []
 a = sum(i, 0)
 for j in i:
  tmp_arr.append(float(j)/float(a))
 norm_conf.append(tmp_arr)
fig = plt.figure()
plt.clf()
ax = fig.add_subplot(111)
ax.set_aspect(1)
res = ax.imshow(np.array(norm_conf), cmap=plt.cm.jet, interpolation="nearest")
width = len(conf_arr)
height = len(conf_arr[0])
for x in xrange(width):
 for y in xrange(height):
  ax.annotate(str(conf_arr[x][y]),  xy=(y,  x),  horizontalalignment="center",
verticalalignment="center")
cb = fig.colorbar(res)
plt.title("Confusion matrix")
plt.ylabel("True label")
plt.xlabel("Predicted label")
plt.xticks(range(width), labels[:width])
plt.yticks(range(height), labels[:height])
plt.savefig(cm_location+"confusion_matrix.png", format="png")
```

Code snippet B.2 This Python code tries to load a trained SVM from a file. If not found, the SVM is trained using provided data (i.e. X_train, Y_train) and saved at svm_location. A prediction is made by evaluating the classifier for feature x_test.

```
from sklearn import svm
from sklearn.externals import joblib
clf = svm.SVC()
try:
  clf = joblib.load(svm_location+"classifier_svm.pkl")
  print "using trained model"
except:
  print "building new model"
  clf.fit(X_train,Y_train)
  joblib.dump(clf, svm_location+"classifier_svm.pkl")
training_score =clf.score(X_train, Y_train)
pred = clf.predict(x_test)
```

Code snippet B.3 This Python code tries to load a trained RF from a file. If not found, the RF is trained using provided data (i.e. X_train, Y_train) and saved at rf_location. A prediction is made by evaluating the classifier for feature x_test.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib
clf = RandomForestClassifier(n_estimators=100)
try:
  clf = joblib.load(rf_location+"classifier_rf.pkl")
  print "using trained model"
except:
  print "building new model"
  clf.fit(X_train,Y_train)
  joblib.dump(clf, rf_location+"classifier_rf.pkl")
training_score =clf.score(X_train, Y_train)
pred = clf.predict(x_test)
```

Code snippet B.4 Semantic segmentation of images using LR in conjunction with the extracted features. Colour coded and confusion matrix results can be saved in files. It contains both Matlab and Python code.

```
%Originally written by Andrew Kae (University of Massachusetts - Amherst)
%Modified by Ionut Gheorghe (Coventry University)
%Confusion matrix is generated in Python by using matpy to call Python from
Matlab (http://algoholic.eu/matpy/)
run('C:/Users/gheorghi/Desktop/JLRDataset/JLRDataset_features/vlfeat-
0.9.19/toolbox/vl_setup');
addpath('model/lr');
addpath('matpy/');

%%% --- default parameter values --- %%%
config_lr;
startup;
%number of segmentation labels
nlabel = 8;

load('sds_large.mat','sds');
%load('esds_large.mat','esds');

fprintf('processing the features!!\n');

load('weights/lr_l2r0.001_rmposfeat0.mat', '-mat', 'w_lr');

verbose = 0;
tot_err = 0;
tot_sp = 0;
tot_err_part = zeros(nlabel, 1);
tot_sp_part = zeros(nlabel, 1);
evaltime = 0;
testList = testnames;
testNums = testnums;

%testList = [testnames validnames];
%testNums = [testnums validnums];
fprintf('total LR test images =  %d \n', length(testList));

gt_splabels_all=[];
pred_all=[];

for i = 1:length(testList),

    % load full data
    gt_casename = sprintf('%s/%s/%s_%04d.dat', gt_dir, testList{i},
testList{i}, testNums(i));
    gt_case = load(gt_casename);
    gt_case = gt_case + 1;
    gt_splabels = gt_case(2:end);   % the first value is the number of nodes
    gt_splabels_all = [gt_splabels_all; gt_splabels];
     % read superpixel features
    [~, H , ~, ~] = getFeatures(testList{i}, testNums(i), features_dir);
    [~, num_sp] = size(H);
    if w_lr.params.rmposfeat,
        H(65:128,:) = [];
    end
```

Code snippet B.4 Continued

```
whiteH = H ./ repmat(sds, [1, num_sp]); % whiten the features
    feat = whiteH';
    numFeat = size(feat, 1);
    feat = [feat ones(numFeat, 1)]; % add bias term
    feat = feat';

    tS = tic;
    labelprob = inference_lr(feat, w_lr.nodeWeights);
    tE = toc(tS);
    evaltime = evaltime + tE;
    average_evaltime = evaltime/i;
    fprintf(' %d tic-toc %d average_evaltime %d\n  ', i, tE, aver-
age_evaltime);

    [~, pred] = max(labelprob ,[], 1);
    pred_all=[pred_all pred];
    err = sum(pred(:) ~= gt_splabels(:));
    tot_err = tot_err + err;
    tot_sp = tot_sp + num_sp;


    for p = 1:nlabel
        tpred = pred(gt_splabels == p);
        tsplabels = gt_splabels(gt_splabels == p);
        err = sum(tpred(:) ~= tsplabels(:));
        tot_err_part(p) = tot_err_part(p) + err;
        tot_sp_part(p) = tot_sp_part(p) + numel(tpred);
    end


     if verbose,
        % acc ?
        fprintf('valid: [%d/%d] err: %d/%d, acc = %g\n', i, length(testList),
err, numFeat, 100*(1-tot_err/tot_sp));
        %load superpixel mat
        supmat_casename = sprintf('%s/%s/%s_%04d.dat', spmat_dir, testList{i},
testList{i}, testNums(i));
        supmat_case = load(supmat_casename);

        %load raw images
        raw_casename = sprintf('%s/%s/%s/_%s_%04d.bmp', lfw_dir, testList{i},
'Rect_left', testList{i}, testNums(i));
        raw_case = imread(raw_casename);

        %load ground truth image
        label_casename = sprintf('%s/%s/%s/_%s_%04d.bmp', label_dir,
testList{i}, 'Rect_left', testList{i}, testNums(i));
        label_case = imread(label_casename);

        %create images directory if it doesn't exist
        if (~exist([imresult_dir '/LR'], 'dir'))
           mkdir([imresult_dir '/LR']);
        end
```

Code snippet B.4 Continued

```
        %location to save results
        results_casename = sprintf('%s/_%s_%04d', [imresult_dir '/LR'],
testList{i}, testNums(i));

        %colorImgWithLabels, visualiuzation and saving
        colorImgWithLabels(supmat_case, raw_case, pred, gt_splabels, la-
bel_case, results_casename);

    else
        if ~mod(i,10),
            fprintf('.');
        end
        if ~mod(i,100),
            fprintf('[%d/%d] ',i,length(testList));
            fprintf('acc = %g\n',100*(1-tot_err/tot_sp));
        end
    end
end

%results generation
%1 grass
%2 tree
%3 sky
%4 dirt
%5 gravel
%6 shrubs
%7 tarmac
%8 void

acc_location=[imresult_dir '/LR_acc/'];
%create cm directory if it doesn't exist
if (~exist(acc_location, 'dir'))
   mkdir(acc_location);
end

fid = fopen([acc_location 'acc.txt'], 'w');
acc = 100*(1-tot_err/tot_sp);
fprintf(fid, 'acc = %g\n', acc);
class_names = {'grass', 'tree', 'sky', 'dirt', 'gravel', 'shrubs', 'tarmac',
'void'};
for p = 1:nlabel
    acc = 100*(1-tot_err_part(p)/tot_sp_part(p));
    fprintf(fid, 'acc of %s = %g\n', class_names{p}, acc);
end
fclose(fid);

predicted=pred_all';
truelabels=gt_splabels_all;
cm_location=[imresult_dir '/LR_cm/'];
%create cm directory if it doesn't exist
if (~exist(cm_location, 'dir'))
   mkdir(cm_location);
end
```

Code snippet B.4 Continued

```
py_export('predicted', 'truelabels', 'cm_location')
stmt= sprintf(['import numpy as np\n'...
'import matplotlib.pyplot as plt\n'...
'from sklearn.metrics import confusion_matrix\n'...
'from sklearn.metrics import accuracy_score\n'...
'Y pred prep= predicted\n'...
'Y_test_prep= truelabels\n'...
'overall_acc= accuracy_score(Y_test_prep, Y_pred_prep)\n'...
'labels= ["grass", "tree", "sky", "dirt", "gravel", "shrubs", "tarmac",
"void"]\n'...
'cm= confusion_matrix(Y_test_prep, Y_pred_prep)\n'...
'conf_arr = cm\n'...
'norm_conf = []\n'...
'for i in conf_arr:\n'...
' a = 0\n'...
' tmp_arr = []\n'...
' a = sum(i, 0)\n'...
' for j in i:\n'...
'  tmp_arr.append(float(j)/float(a))\n'...
' norm_conf.append(tmp_arr)\n'...
'fig = plt.figure()\n'...
'plt.clf()\n'...
'ax = fig.add_subplot(111)\n'...
'ax.set_aspect(1)\n'...
'res = ax.imshow(np.array(norm_conf), cmap=plt.cm.jet, interpola-
tion="nearest")\n'...
'width = len(conf_arr)\n'...
'height = len(conf_arr[0])\n'...
'for x in xrange(width):\n'...
' for y in xrange(height):\n'...
'  ax.annotate(str(conf_arr[x][y]), xy=(y, x), horizontalalignment="center",
verticalalignment="center")\n'...
'cb = fig.colorbar(res)\n'...
'plt.title("Confusion matrix LR")\n'...
'plt.ylabel("True label")\n'...
'plt.xlabel("Predicted label")\n'...
'plt.xticks(range(width), labels[:width])\n'...
'plt.yticks(range(height), labels[:height])\n'...
'plt.savefig(cm_location+"confusion_matrix_LR.png", format="png")']);
py('eval', stmt)
py_import('overall_acc')
```

Code snippet B.5 Semantic segmentation of images using CRF in conjunction with the extracted features. Colour coded and confusion matrix results can be saved in files. It contains both Matlab and Python code.

```
%Originally written by Andrew Kae (University of Massachusetts - Amherst)
%Modified by Ionut Gheorghe (Coventry University)
%Confusion matrix is generated in Python by using matpy to call Python from
Matlab (http://algoholic.eu/matpy/)
run('C:/Users/gheorghi/Desktop/JLRDataset/JLRDataset_features/vlfeat-
0.9.19/toolbox/vl_setup');
addpath('model/crf');
addpath('matpy/');

%%% --- default parameter values --- %%%
config_crf;
startup;
nlabel = 8;     % number of segmentation labels

load('sds_large.mat','sds');
load('esds_large.mat','esds');

fprintf('processing the features!!\n');

load('weights/lr_l2r0.001_rmposfeat0/crf_l2n0.001_l2e0.0001_rmposfeat0.mat',
'-mat', 'w_crf');

verbose = 1;
tot_err = 0;
tot_sp = 0;
tot_err_part = zeros(nlabel, 1);
tot_sp_part = zeros(nlabel, 1);
evaltime = 0;
testList = testnames;
testNums = testnums;
%testList = [testnames validnames];
%testNums = [testnums validnums];
fprintf('total CRF test images =  %d \n', length(testList));

gt_splabels_all=[];
pred_all=[];

for i = 1:length(testList),

    % load full data
    gt_casename = sprintf('%s/%s/%s_%04d.dat', gt_dir, testList{i},
testList{i}, testNums(i));
    gt_case = load(gt_casename);
    gt_case = gt_case + 1;
    gt_splabels = gt_case(2:end);   % the first value is the number of nodes
    gt_splabels_all = [gt_splabels_all; gt_splabels];
     % read superpixel features
    [numNodes, H , E, S] = getFeatures(testList{i}, testNums(i), fea-
tures_dir);
    X = struct('numNodes', numNodes, 'adjmat', {E}, 'nodeFeatures', {H},
'edgeFeatures', {S});
    [~, num_sp] = size(X.nodeFeatures);
```

Code snippet B.5 Continued

```
% scale features
    if w_crf.params.rmposfeat,
        X.nodeFeatures(65:128,:) = [];
    end
    X.nodeFeatures = bsxfun(@rdivide,X.nodeFeatures,sds);
    X.nodeFeatures(end+1,:) = 1;

    [xe, ye] = find(X.adjmat > 0);
    for j=1:length(xe)
        X.edgeFeatures{xe(j),ye(j)} = X.edgeFeatures{xe(j),ye(j)} ./ esds;
        X.edgeFeatures{xe(j),ye(j)}(end+1) = 1;
    end

    tS = tic;
    labelprob = inference_crf(w_crf, X, nlabel, 0);
    tE = toc(tS);
    evaltime = evaltime + tE;
    average_evaltime = evaltime/i;
    fprintf(' %d tic-toc %d average_evaltime %d\n  ', i, tE, aver-
age_evaltime);

    [~, pred] = max(labelprob ,[], 1);
    pred_all=[pred_all pred];
    err = sum(pred(:) ~= gt_splabels(:));
    tot_err = tot_err + err;
    tot_sp = tot_sp + num_sp;


    for p = 1:nlabel
        tpred = pred(gt_splabels == p);
        tsplabels = gt_splabels(gt_splabels == p);
        err = sum(tpred(:) ~= tsplabels(:));
        tot_err_part(p) = tot_err_part(p) + err;
        tot_sp_part(p) = tot_sp_part(p) + numel(tpred);
    end


     if verbose,
        % acc ?
        % fprintf('valid: [%d/%d] err: %d/%d, acc = %g\n', i,
length(testList), err, numFeat, 100*(1-tot_err/tot_sp));
        %load superpixel mat
        supmat_casename = sprintf('%s/%s/%s_%04d.dat', spmat_dir, testList{i},
testList{i}, testNums(i));
        supmat_case = load(supmat_casename);

        %load raw images
        raw_casename = sprintf('%s/%s/%s/_%s_%04d.bmp', lfw_dir, testList{i},
'Rect_left', testList{i}, testNums(i));
        raw_case = imread(raw_casename);

        %load ground truth image
        label_casename = sprintf('%s/%s/%s/_%s_%04d.bmp', label_dir,
testList{i}, 'Rect_left', testList{i}, testNums(i));
        label_case = imread(label_casename);
```

Code snippet B.5 Continued

```
%create images directory if it doesn't exist
        if (~exist([imresult_dir '/CRF'], 'dir'))
           mkdir([imresult_dir '/CRF']);
        end

        %location to save results
        results_casename = sprintf('%s/_%s_%04d', [imresult_dir '/CRF'],
testList{i}, testNums(i));

        %colorImgWithLabels, visualiuzation and saving
        colorImgWithLabels(supmat_case, raw_case, pred, gt_splabels, la-
bel_case, results_casename);
    else
        if ~mod(i,10),
            fprintf('.');
        end
        if ~mod(i,100),
            fprintf('[%d/%d] ',i,length(testList));
            fprintf('acc = %g\n',100*(1-tot_err/tot_sp));
        end
     end
end

%results generation
%1 grass
%2 tree
%3 sky
%4 dirt
%5 gravel
%6 shrubs
%7 tarmac
%8 void

acc_location=[imresult_dir '/CRF_acc/'];
%create cm directory if it doesn't exist
if (~exist(acc_location, 'dir'))
   mkdir(acc_location);
end

fid = fopen([acc_location 'acc.txt'], 'w');
acc = 100*(1-tot_err/tot_sp);
fprintf(fid, 'acc = %g\n', acc);
class_names = {'grass', 'tree', 'sky', 'dirt', 'gravel', 'shrubs', 'tarmac',
'void'};
for p = 1:nlabel
    acc = 100*(1-tot_err_part(p)/tot_sp_part(p));
    fprintf(fid, 'acc of %s = %g\n', class_names{p}, acc);
end
fclose(fid);
predicted=pred_all';
truelabels=gt_splabels_all;
cm_location=[imresult_dir '/CRF_cm/'];
%create cm directory if it doesn't exist
if (~exist(cm_location, 'dir'))
   mkdir(cm_location);
end
```

Code snippet B.5 Continued

```
 py_export('predicted', 'truelabels', 'cm_location')
stmt= sprintf(['import numpy as np\n'...
'import matplotlib.pyplot as plt\n'...
'from sklearn.metrics import confusion_matrix\n'...
'from sklearn.metrics import accuracy_score\n'...
'Y_pred_prep= predicted\n'...
'Y_test_prep= truelabels\n'...
'overall_acc= accuracy_score(Y_test_prep, Y_pred_prep)\n'...
'labels= ["grass", "tree", "sky", "dirt", "gravel", "shrubs", "tarmac",
"void"]\n'...
'cm= confusion_matrix(Y_test_prep, Y_pred_prep)\n'...
'conf_arr = cm\n'...
'norm_conf = []\n'...
'for i in conf_arr:\n'...
' a = 0\n'...
' tmp_arr = []\n'...
' a = sum(i, 0)\n'...
' for j in i:\n'...
'  tmp_arr.append(float(j)/float(a))\n'...
' norm_conf.append(tmp_arr)\n'...
'fig = plt.figure()\n'...
'plt.clf()\n'...
'ax = fig.add_subplot(111)\n'...
'ax.set_aspect(1)\n'...
'res = ax.imshow(np.array(norm_conf), cmap=plt.cm.jet, interpola-
tion="nearest")\n'...
'width = len(conf_arr)\n'...
'height = len(conf_arr[0])\n'...
'for x in xrange(width):\n'...
' for y in xrange(height):\n'...
'  ax.annotate(str(conf_arr[x][y]), xy=(y, x), horizontalalignment="center",
verticalalignment="center")\n'...
'cb = fig.colorbar(res)\n'...
'plt.title("Confusion matrix CRF")\n'...
'plt.ylabel("True label")\n'...
'plt.xlabel("Predicted label")\n'...
'plt.xticks(range(width), labels[:width])\n'...
'plt.yticks(range(height), labels[:height])\n'...
'plt.savefig(cm_location+"confusion_matrix_CRF.png", format="png")']);
py('eval', stmt)
py_import('overall_acc')
```

Code snippet B.6 This code is used as part of the feature extraction process (e.g. FPFH) using Point Cloud library and Triclops application program interface. More specifically it is used to build a mex file that can be called as a function from within Matlab

```
#include "mex.h"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "pcl/io/pcd_io.h"
#include "pcl/point_types.h"
#include "pcl/visualization/cloud_viewer.h"
#include "pcl/features/normal_3d.h"
#include "pcl/features/fpfh.h"
#include "pcl/filters/voxel_grid.h"
#include "triclops.h"


//triclopsXYZToRCD -- Converts true 3D points into image coordinates.

// Print error and quit program
#define _HANDLE_TRICLOPS_ERROR( description, error ){ \
if( error != TriclopsErrorOk ) { \
        mexPrintf( "*** Triclops Error '%s' at line %d :\n\t%s\n", \
        triclopsErrorToString( error ), \
        __LINE__, description ); mexErrMsgTxt("An Error there was"); \
} }

void ExitFcn(void);
void DoInit(void);
void MakeFPFHImage(  mxArray *plhs[] , pcl::PointCloud<pcl::PointXYZRGB>::Ptr
cloudColor, pcl::PointCloud<pcl::FPFHSignature33>::Ptr descriptors_FPFH );
static bool inited = false;
static int currres;
static mwSize OutDims[2];
static TriclopsContext      triclops;
static TriclopsError          te;
static int outCol; static int outRow;
static int minDisp; static int maxDisp;

int rhsparse_new=0;

void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[] )
{

       char *cmd;
       bool CommandAccepted = false;

     // Object for storing the original large cloud
       pcl::PointCloud<pcl::PointXYZRGB>::Ptr cloud (new
pcl::PointCloud<pcl::PointXYZRGB>);
     // Object for storing the FPFH descriptors for each point.
     pcl::PointCloud<pcl::FPFHSignature33>::Ptr descriptors_FPFH(new
pcl::PointCloud<pcl::FPFHSignature33>());
     // Object for storing the normals.
       pcl::PointCloud<pcl::Normal>::Ptr normals(new
pcl::PointCloud<pcl::Normal>);


       rhsparse_new = 0;
```

Code snippet B.6 Continued

```
if(nrhs<1) mexErrMsgTxt("Need command string input");
     if (!mxIsChar(prhs[rhsparse_new]))   mexErrMsgTxt("Unrecognised command
or none provided");
     cmd = mxArrayToString(prhs[rhsparse_new++]); //get command string

       if (!strcmp(cmd, "workit"))
       {
        CommandAccepted = true;
            //this is a pointer to the entire cloud data
            int nrows=(int) mxGetM(prhs[1]); // M is number of rows
            int ncols=(int) mxGetN(prhs[1]); //

            float* data=(float*)mxGetData(prhs[1]);

            float x, y, z;
            unsigned char pr, pg, pb;
            pcl::PointXYZRGB point;
        std::cout<<" Points # "<<nrows<<std::endl;
        //std::cout<<"ncols: "<<ncols<<std::endl;
        int pointn = nrows;

            for (int i=0; i< pointn; i++)
            {
                  x=  data[i];
                  y=  data[i+1*nrows];
                  z=  data[i+2*nrows];
                  pr= data[i+3*nrows];
                  pg= data[i+4*nrows];
                  pb= data[i+5*nrows];
            point.x=x;
                  point.y=y;
                  point.z=z;
                  uint32_t rgb = (static_cast<uint32_t>(pr) << 16 |
            static_cast<uint32_t>(pg) << 8 | static_cast<uint32_t>(pb));
            point.rgb = *reinterpret_cast<float*>(&rgb);
            cloud->points.push_back (point);

            }
       // Downsampling the cloud to speed up the computation
       pcl::PointCloud<pcl::PointXYZRGB>::Ptr cloudColor (new
pcl::PointCloud<pcl::PointXYZRGB>);
       pcl::VoxelGrid<pcl::PointXYZRGB> sor;
     sor.setInputCloud (cloud);
     sor.setLeafSize (0.1f, 0.1f, 0.1f);
     sor.filter (*cloudColor);
     std::cout<<" Downsampling done !"<<std::endl;
     std::cout<<" Have # "<<cloudColor->size()<<std::endl;

     // Estimate the normals.
       pcl::NormalEstimation<pcl::PointXYZRGB, pcl::Normal> normalEstimation;
       normalEstimation.setInputCloud(cloudColor);
       normalEstimation.setRadiusSearch(0.3);
       pcl::search::KdTree<pcl::PointXYZRGB>::Ptr kdtree(new
pcl::search::KdTree<pcl::PointXYZRGB>);
       normalEstimation.setSearchMethod(kdtree);
       normalEstimation.compute(*normals);
```

Code snippet B.6 Continued

```
// FPFH estimation object.
       pcl::FPFHEstimation<pcl::PointXYZRGB, pcl::Normal,
pcl::FPFHSignature33> fpfh;
       fpfh.setInputCloud(cloudColor);
       fpfh.setInputNormals(normals);
       fpfh.setSearchMethod(kdtree);
       // Search radius, to look for neighbors. Note: the value given here has
to be
       // larger than the radius used to estimate the normals.
       fpfh.setRadiusSearch(0.5);
       fpfh.compute(*descriptors_FPFH);
     std::cout<<" FPFH descriptors done !"<<std::endl;
     // std::cout<<" Found # "<<descriptors_FPFH->size()<<std::endl;

     DoInit();
     MakeFPFHImage(plhs, cloudColor, descriptors_FPFH );

     mxFree (cmd);
     return;

       }

     if (!CommandAccepted)
       {
        mexErrMsgTxt("Unknown command");
     }

 mxFree(cmd);

}

void MakeFPFHImage(  mxArray *plhs[] , pcl::PointCloud<pcl::PointXYZRGB>::Ptr
cloudColor, pcl::PointCloud<pcl::FPFHSignature33>::Ptr descriptors_FPFH )
{
     //p, q, r as x, y, z
     float p, q, r, row, col, disp;
     int red, green, blue;
     int dptr=-1;
     float *Out3dPtr;
     void *newptr;
     Out3dPtr= (float*) mxMalloc((5 + 33)*480*640*sizeof(float)); // {row,
col, red, green, blue}*rows*cols
     for (int i = 0; i <cloudColor->size(); i++)
     {
       //code here using triclops
         p= cloudColor->points[i].x;
         q= cloudColor->points[i].y;
         r= cloudColor->points[i].z;
         red= cloudColor->points[i].r;
         green= cloudColor->points[i].g;
         blue= cloudColor->points[i].b;

         triclopsXYZToRCD(triclops, p, q, r, &row, &col, &disp);

         dptr++; Out3dPtr[dptr] = (int)row;
         dptr++; Out3dPtr[dptr] = (int)col;
         dptr++; Out3dPtr[dptr] = red;
         dptr++; Out3dPtr[dptr] = green;
         dptr++; Out3dPtr[dptr] = blue;
```

Code snippet B.6 Continued

```
    for (int pos = 0; pos < descriptors_FPFH->points[i].descriptorSize();
pos++)
          {
              dptr++; Out3dPtr[dptr] = descriptors_FPFH-
>points[i].histogram[pos];
          }

      }

    plhs[0] = mxCreateNumericMatrix(0, 0, mxSINGLE_CLASS, mxREAL);
    newptr = mxRealloc(Out3dPtr, 480*640*(5 + 33)*sizeof( float));
    mxSetData(plhs[0], newptr);
    mxSetM(plhs[0], (5 + 33));
    mxSetN(plhs[0], 480*640);

}

void DoInit(void)
{

    //minDisp = 0;
    //maxDisp = 160;
    outRow = 480;
    outCol = 640;

    char* szCalFile = "bumblebee11123884-current.cal";
    // Get the camera calibration data
    te = triclopsGetDefaultContextFromFile( &triclops, szCalFile );
    _HANDLE_TRICLOPS_ERROR( "triclopsGetDefaultContextFromFile(): Can't open
calibration file", te );
    // set output resolution to input resolution
    triclopsSetDisparity( triclops, minDisp, maxDisp );
    triclopsSetResolution( triclops, outRow,  outCol ) ;
    triclopsSetStereoMask( triclops, 11 );
    triclopsSetSubpixelInterpolation( triclops, true);
    triclopsSetTextureValidation( triclops, false);

    OutDims[0]=(outCol); //always return single row matrix.
    OutDims[1]=(outRow); //setup appropriate output matrix size
    currres = outRow;
    mexAtExit( ExitFcn );

    // _HANDLE_FLYCAPTURE_ERROR( "flycaptureInitialize()", fe );
    inited = true;

}

void ExitFcn(void)
{
    te = triclopsDestroyContext( triclops );
    //fe = flycaptureDestroyContext( flycapture );
    _HANDLE_TRICLOPS_ERROR( "triclopsDestroyContext()", te );
}
```

Code snippet B.7 Semantic segmentation of images using GLOC in conjunction with the extracted features. Colour coded and confusion matrix results can be saved in files. It contains both Matlab and Python code.

```
%Originally written by Andrew Kae (University of Massachusetts - Amherst)
%Modified by Ionut Gheorghe (Coventry University)
%Confusion matrix is generated in Python by using matpy to call Python from
Matlab (http://algoholic.eu/matpy/)
run('C:/Users/gheorghi/Desktop/JLRDataset/JLRDataset_features/vlfeat-
0.9.19/toolbox/vl_setup');
addpath('model/crbm');
addpath('model/gloc');
addpath('matpy/');

%%% --- default parameter values --- %%%
config_gloc;
startup;        %
nlabel = 2;     % number of segmentation labels

load('sds_large.mat','sds');
load('esds_large.mat','esds');

fprintf('processing the features!!\n');

load('weights/slr_l2r0.0001_rmposfeat1_N16/gloc_LFW_nD429_nL2_N16_l2n0.0001_l2
e0.0001_rbm_R24_nH400_l2r0.0001_eps0.002308_CD30_ann0_bS289_0300.mat', '-mat',
'w_gloc');


%slr_l2r0.0001_rmposfeat1_N16/gloc_LFW_nD429_nL2_N16_l2n0.0001_l2e0.0001_rbm_R
24_nH400_l2r0.0001_eps0.002308_CD30_ann0_bS289_0300.mat
             %for heightons

%slr_l2r0.0001_rmposfeat1_N16/gloc_LFW_nD665_nL2_N16_l2n0.0001_l2e0.0001_rbm_R
24_nH400_l2r0.0001_eps0.002308_CD30_ann0_bS289_0300.mat
             %for FPFHtons

w_gloc.vishidrs = re-
shape(w_gloc.vishid,size(w_gloc.vishid,1)*size(w_gloc.vishid,2),size(w_gloc.vi
shid,3));
w_gloc.visbiasrs  = re-
shape(w_gloc.visbiases,size(w_gloc.vishid,1)*size(w_gloc.vishid,2),1);
w_gloc.nodeWeightsrs = re-
shape(w_gloc.nodeWeights,size(w_gloc.nodeWeights,1)*size(w_gloc.nodeWeights,2)
,size(w_gloc.nodeWeights,3));
w_gloc.vishidperm = re-
shape(w_gloc.vishid,size(w_gloc.vishid,1),size(w_gloc.vishid,2)*size(w_gloc.vi
shid,3));

verbose = 1;
tot_err = 0;
tot_sp = 0;
tot_err_part = zeros(nlabel, 1);
tot_sp_part = zeros(nlabel, 1);
evaltime = 0;
testList = testnames;
testNums = testnums;
```

Code snippet B.7 Continued

```
fprintf('total GLOC test images =  %d \n', length(testList));

gt_splabels_all=[];
pred_all=[];

sds(65:128) = [];

for i = 1:length(testList),

    % load full data
    gt_casename = sprintf('%s/%s_%06d.dat', gt_dir, testList{i}, testNums(i));
    gt_case = load(gt_casename);
    gt_case = gt_case + 1;
    gt_splabels = gt_case(2:end);   % the first value is the number of nodes
    gt_splabels_all = [gt_splabels_all; gt_splabels];

     % read superpixel features
    %[numNodes, H , E, S] = getFeatures(testList{i}, testNums(i), fea-
tures_dir);
    [numNodes, H, E, S] = getFeatures(testList{i}, testNums(i), features_dir);
    %w_gloc.params.rmposfeat

    if w_gloc.params.rmposfeat,
        H(65:128, :) = [];
    end

    % node features
    H = bsxfun(@rdivide,H,sds);
    H(end+1,:) = 1; % add bias term

    % edge features
    [xe, ye] = find(E > 0);
    for j = 1:length(xe),
        S{xe(j),ye(j)} = S{xe(j),ye(j)} ./ esds;
        S{xe(j),ye(j)}(end+1) = 1; % add bias term
    end
    X = struct('numNodes', numNodes, 'adjmat', {E}, 'nodeFeatures', {H},
'edgeFeatures', {S});
    num_sp = numNodes;
    % read superpixel data
    spfile = sprintf('%s/%s_%06d.dat', spmat_dir, testList{i}, testNums(i));
    sp = load(spfile) + 1;

    %load raw images to find size
    raw_casename = sprintf('%s/%s_%06d.png', lfw_dir, testList{i}, test-
Nums(i));
    raw_case = imread(raw_casename);
    sz = size(raw_case);
    olddimy = sz(1);
    olddimx = sz(2);
```

Code snippet B.7 Continued

```
% read projection matrix
    [~, proj_sp] = create_mapping(sp,num_sp,sqrt(w_gloc.params.numNodes_crf),
olddimy, olddimx);
    proj_crf = proj_sp;

    w_gloc.params.numNodes_crf

    % projection matrices
    [proj_blk, ~] = create_mapping(sp,num_sp,sqrt(w_gloc.params.numNodes_rbm),
olddimy, olddimx);
    proj_rbm = proj_blk;

    w_gloc.params.numNodes_rbm

    tS = tic;
    labelprob = inference_gloc(X, w_gloc, w_gloc.params, proj_crf, proj_rbm);
    tE = toc(tS);
    evaltime = evaltime + tE;
    average_evaltime = evaltime/i;
    fprintf(' %d tic-toc %d average_evaltime %d\n  ', i, tE, aver-
age_evaltime);

    [~, pred] = max(labelprob ,[], 1);
    pred_all=[pred_all pred];
    err = sum(pred(:) ~= gt_splabels(:));
    tot_err = tot_err + err;
    tot_sp = tot_sp + num_sp;

    for p = 1:nlabel
        tpred = pred(gt_splabels == p);
        tsplabels = gt_splabels(gt_splabels == p);
        err = sum(tpred(:) ~= tsplabels(:));
        tot_err_part(p) = tot_err_part(p) + err;
        tot_sp_part(p) = tot_sp_part(p) + numel(tpred);
    end


     if verbose,
        % acc ?
        % fprintf('valid: [%d/%d] err: %d/%d, acc = %g\n', i,
length(testList), err, numFeat, 100*(1-tot_err/tot_sp));
        %load superpixel mat
        supmat_casename = sprintf('%s/%s_%06d.dat', spmat_dir, testList{i},
testNums(i));
        supmat_case = load(supmat_casename);

        %load raw images
        raw_casename = sprintf('%s/%s_%06d.png', lfw_dir, testList{i}, test-
Nums(i));
        raw_case = imread(raw_casename);
        %size(raw_case)
```

Code snippet B.7 Continued

```
        %load ground truth image
        %sprintf(fn, "%s/%s/%s%s_%s_%06d.png", label_dir.c_str(), "testing",
"gt_" ,  s.substr(8, 11).c_str(), "road", n);
        class_name = 'road';
        category = 'testing/gt_image_2';
        label_casename = sprintf('%s%s/%s_%s_%06d.png', label_dir ,category,
testList{i}(16:18), class_name, testNums(i));
        label_case = imread(label_casename);

        %size(label_case)

        %create images directory if it doesn't exist
        if (~exist([imresult_dir '/GLOC'], 'dir'))
           mkdir([imresult_dir '/GLOC']);
        end
        if (~exist([imresult_dir '/GLOC/testing/'], 'dir'))
           mkdir([imresult_dir '/GLOC/testing/']);
        end

        if (~exist([imresult_dir '/GLOC/testing/image_2/'], 'dir'))
           mkdir([imresult_dir '/GLOC/testing/image_2/']);
        end

        %location to save results
        results_casename = sprintf('%s/%s_%06d', [imresult_dir '/GLOC'],
testList{i}, testNums(i));

        %colorImgWithLabels, visualiuzation and saving
        colorImgWithLabels(supmat_case, raw_case, pred, gt_splabels, la-
bel_case, results_casename);
    else
        if ~mod(i,10),
            fprintf('.');
        end
        if ~mod(i,100),
            fprintf('[%d/%d] ',i,length(testList));
            fprintf('acc = %g\n',100*(1-tot_err/tot_sp));
        end
     end
end
```

Code snippet B.7 Continued

```
%results generation
%1 grass
%2 tree
%3 sky
%4 dirt
%5 gravel
%6 shrubs
%7 tarmac
%8 void

acc_location=[imresult_dir '/GLOC_acc/'];
%create cm directory if it doesn't exist
if (~exist(acc_location, 'dir'))
   mkdir(acc_location);
end

fid = fopen([acc_location 'acc.txt'], 'w');
acc = 100*(1-tot_err/tot_sp);
fprintf(fid, 'acc = %g\n', acc);
class_names = {'void', 'road'};
for p = 1:nlabel
    acc = 100*(1-tot_err_part(p)/tot_sp_part(p));
    fprintf(fid, 'acc of %s = %g\n', class_names{p}, acc);
end
fclose(fid);

predicted=pred_all';
truelabels=gt_splabels_all;
cm_location=[imresult_dir '/GLOC_cm/'];
%create cm directory if it doesn't exist
if (~exist(cm_location, 'dir'))
   mkdir(cm_location);
end
py_export('predicted', 'truelabels', 'cm_location')
stmt= sprintf(['import numpy as np\n'...
'import matplotlib.pyplot as plt\n'...
'from sklearn.metrics import confusion_matrix\n'...
'from sklearn.metrics import accuracy_score\n'...
'Y_pred_prep= predicted\n'...
'Y_test_prep= truelabels\n'...
'overall_acc= accuracy_score(Y_test_prep, Y_pred_prep)\n'...
'labels= ["void", "road"]\n'...
'cm= confusion_matrix(Y_test_prep, Y_pred_prep)\n'...
'conf_arr = cm\n'...
'norm_conf = []\n'...
'for i in conf_arr:\n'...
' a = 0\n'...
' tmp_arr = []\n'...
' a = sum(i, 0)\n'...
' for j in i:\n'...
'  tmp_arr.append(float(j)/float(a))\n'...
' norm_conf.append(tmp_arr)\n'...
```

Code snippet B.7 Continued

```
'fig = plt.figure()\n'...
'plt.clf()\n'...
'ax = fig.add_subplot(111)\n'...
'ax.set_aspect(1)\n'...
'res = ax.imshow(np.array(norm_conf), cmap=plt.cm.jet, interpola-
tion="nearest")\n'...
'width = len(conf_arr)\n'...
'height = len(conf_arr[0])\n'...
'for x in xrange(width):\n'...
' for y in xrange(height):\n'...
'  ax.annotate(str(conf_arr[x][y]), xy=(y, x), horizontalalignment="center",
verticalalignment="center")\n'...
'cb = fig.colorbar(res)\n'...
'plt.title("Confusion matrix GLOC")\n'...
'plt.ylabel("True label")\n'...
'plt.xlabel("Predicted label")\n'...
'plt.xticks(range(width), labels[:width])\n'...
'plt.yticks(range(height), labels[:height])\n'...
'plt.savefig(cm_location+"confusion_matrix_GLOC.png", format="png")']);
py('eval', stmt)
py import('overall acc')
```

Appendix C Data samples

Sample images C.1 The JLR dataset

Sample images C.1 Continued

Appendix C Data samples

[redacted]

Sample images C.2 Continued

Appendix D Published work

Superpixel based semantic segmentation for assistance in varying terrain driving conditions

Ionut Gheorghe, Weidong Li, Thomas Popham and Keith J. Burnham

Control Theory and Applications Centre, Coventry University, Coventry CV1 5FB, UK
gheorghi@uni.coventry.ac.uk
{aa3719, ctac}@coventry.ac.uk
Jaguar & Land Rover Research, University of Warwick, Coventry CV4 7AL, UK
tpopham@jaguarlandrover.com

Abstract. Vehicle drivability and maneuverability can be improved by increasing the environment awareness via sensory inputs. In particular, off-road capable vehicles possess subsystems which are configurable to the driving conditions. In this work, a vision solution is explored as a precursor to autonomous toggling between different operating modes. The emphasis is on selecting an appropriate response to transitions from one terrain type to another. Given a forward facing camera, images are partitioned into pixel subsets known as superpixels in order to be classified. The quality of this semantic segmentation is considered for classes such as {grass, tree, sky, tarmac, dirt, gravel, shrubs}. Colour and texture are combined together to form visual cues and address this image recognition problem with good segmentation results.

Keywords: Terrain classification, semantic segmentation, superpixels, texture, colour, machine learning

Key learning features as means for terrain classification

Ionut Gheorghe, Weidong Li, Thomas Popham, Anna Gaszczak and Keith Burnham

Control Theory and Applications Centre, Coventry University, Coventry CV1 5FB, UK
gheorghi@uni.coventry.ac.uk
{aa3719, ctac}@coventry.ac.uk
Jaguar & Land Rover Research, University of Warwick, Coventry CV4 7AL, UK
{tpopham, agaszcza}@jaguarlandrover.com

Abstract. Modern vehicles seek autonomous subsystems adaptability to ever-changing terrain types in pursuit of enhanced drivability and maneuverability. The impact of key features on the classification accuracy of terrain types using a colour camera is investigated. A handpicked combination of texture and colour as well as a simple unsupervised feature representation is proposed. Although the results are restricted to only four classes {grass, tarmac, dirt, gravel} the learned features can be tailored to suit more classes as well as different scenarios altogether. The novel aspect stems from the feature representation itself as a global gist for three quantities of interest within each image: background, foreground and noise. In addition to that, the frequency affinity of the Gabor wavelet gist component to perspective images is mitigated by inverse homography mapping. The emphasis is thus on feature selection in an unsupervised manner and a framework for integrating learned features with standard off the shelf machine learning algorithms is provided. Starting with a colour hue and saturation histogram as fundamental building block, more complex features such as GLCM, k-means and GMM quantities are gradually added to observe their integrated effect on class prediction for three parallel regions of interest. The terrain classification problem is tackled with promising results using a forward facing camera.

Keywords: Terrain classification, machine learning, gist, GLCM, texture, colour, homography

Institute of Measurement and Control
Measurement+Control
The Journal of the Institute of Measurement and Control

Urban traffic simulators for intelligent transportation systems

Olivier Haas, Coventry University, Coventry, CV1 2TL, UK
Shoaib Kamran, Airwave Solutions Ltd, Rugby, CV23 0PD

Pawel Jaworski, Coventry University, Coventry, CV1 2TL, UK
Ionut Gheorghe, Coventry University, Coventry, CV1 2TL, UK

o.haas@coventry.ac.uk (corresponding Author)

Abstract

This paper describes the intelligent transportation systems (ITS) technologies, methods, components and their application to traffic simulation and management. Examples of an agent based ITS mescopic simulator and a cloud based microscopic simulator are used to illustrate urban traffic management and incident response applications.